

WIZnet

©2006 WIZnet Co., Inc. All Rights Reserved.

For more information, visit our website at http://www.wiznet.co.kr



# **Document History Information**

Revision	Data	Description
Ver. 1.0	December 7, 2006	Original Document



# **WIZnet's Online Technical Support**

If you have something to ask about WIZnet Products, Write down your question on <u>Q&A Board</u> in WIZnet website (<u>www.wiznet.co.kr</u>). WIZnet Engineer will give an answer as soon as possible.

WIZNEL	Products Techno	logy Support	How to buy	News P	artners A	bout us
	A 08A	(Em) FAQ Lead-	free Status Dov	mioad Related	Link	
	Click			1.11	3	
If you find the bes	st network solution, her	e is WIZnet.			A The	•
WIZnet provides the s using memory-like TC	implest and smartest Internet P/IP chips and component n	connectivity nodules	WIZne		The surger	
			cosoi	-	1	
			2	Qua		
Products	- products nam	e Iv	MORE	Real H	hy ardwired	
iinChip <sup>19</sup> / NM7000A 😆 Home		( do mo)		T	CP/IP?	
TCP/IP Ethernet Module NM7000A is the mini network modu TCP/IP chip). Ethernet PHY and oth	le including W3100A(hardwired er olue logice, it can be used	i and			iy WiZnet tech is ique & competitiv	e?
				T	DE / IPv6 Cor	е
IM7100A Evaluation Board	NJ MURE	The second			Y	
NM7000A is the mini network modu TCP/IP chip), Ethernet PHY and oth	le including W3100A(hardwired er glue logics. It can be used			Lead - free W3100A W3100A-	F How	v to ome a net Partner?
WIZnet News			MORE	-		+
	12 001 20 00			WIZnet Downloa Center		ure of Inet A
2005-04-12 WIZnet Office F	(elocation					
TECHNICAL SI	JPPORT		4	IMEL Ci	Chip Ment	edded interne xit OS
VVIZnet Copyright (c) 2 Address:#306	005 WiZnet Co., Inc., All rights rese Advanced Tech. RSD Center, 68 Y	rved. step-dong Bundeng-gu S	ungnam City, Kyun	ggi-do 463-816 Ko	rea	
				HOME • SITEMAP	• CONTACT US	• KOREAN
WIZnet	Products Technol	logy Support	How to buy	HOME SITEMAP News F	• CONTACT US	• KOREAN
WIZnet	Products Techno	logy Support	How to buy	HOME SITEMAP	• CONTACT US	• KOREAN
WIZnet	Products Techno	ology Support	How to buy	HOME SITEMAP	• CONTACT US	• KOREAN
WIZnet	Products Techno	ology Support	How to buy	Nows F	+ CONTACT UB Partners	<ul> <li>KOREAN</li> <li>About us</li> <li>Znet. 9</li> </ul>
WIZnet	Products Techno	ology Support	How to buy the best networks to simplest <i>i-like TCP/IP ch</i>	Nows F Nows F rock solution nd smartest Internet	• CONTACT US	• KOREAN About us Znet.
WIZnet SUPPORT	Products Technol O Q&A (En)	Nogy Support	How to buy the best networks due the simplest effect (CP) IP ch	Nows F Nows F rock solution	CONTACT US     CONTACT US     Artners     A     here is WI2     former competition	• KOREAN About us Znet. 9
WIZnet	Products Technol • Q&A(En)	ology Support	How to buy the best networks the the simplest dist the simplest	Nows F Nows F rock solution of smartest Inter-	CONTACT UE     Arthors     A     here is WIZ     more is WIZ     home > support	• KOREAN Ibout us Znet. 9 T > OSA
WIZnet SUPPORT WZnetak	Products Technol Q&A(En)	ology Support	How to buy the best networks the the simplest -fike TCP/IP ch	Nows F Nows F North Solution	CONTACT US     CONTACT US     Contract US	<ul> <li>KOREAN</li> <li>Nobul us</li> <li>Inet.</li> <li>T &gt; QSA</li> <li>HTS</li> </ul>
WIZnet SUPPORT WZnetak GAAKKY GAAKKY FAQ	Products Technol Q&A(En) No   45 Senia configuration	Vegy Support	How to buy	Nows F Nows F Nows A Nows A Nows A Nows A Nows A Nows A Nows A	CONTACT US     CONTACT US     Cartners     A     Contract US	× KOREANU US Libout US Znet. 9 9 17 > Q&A HITS 110
WIZnet WZneth Construction Cons	Products Technol Products Technol Q&A(En) Ro 45 Setial configuration 45 How chect that Tx b	Alogy Support	How to buy	Nows F Nows F ork solution nd smartel http: pip and component NAME L Vtyusha	CONTACT US     CONTACT US     Cartners     A     Contract US	• KOREAN About us Znet. 9 9 7 7 > 064 HTS 110 105
WIZnet         WZNetAu         OKARON         OKARON         OKARON         PAQ         Lead-free Status         Download	Products Technol Products Technol Q&A(En)	Alogy Support Support Witzer provi using memory SUBJECT rtm210_(0) utter is en(0) ion (0)	How to buy	Nows F Nows F ork solution nd smartel http: pip and composi- pip and composi- tip and composi- pip and composi- tip and composi- tip and composi- pip and composi- tip and compo	CONTACT US     Contract     Contract US     Contract US     Contract US	• KOREAN hbout us Znet. • • • • • • • • • • • • • • • • • • •
WIZnet         Wiznetak         Other         Other         Other         Other         Other         Other         Other         Operations         Operations      <	Products Technol Products Technol Q&A(En)  Q&A(En)  No 45 Senial configuration 44 How chect that Tx b 43 APt for AVR Codevis 42 heaving problem (0)	dogy Support	How to buy	News F News F ork solution of solution of snarres Inter- ps and composi- solution y and composi- y and	CONTACT US     Contract US     Cartners     A     Contract US     Contract US     Contract     Contract	• KOREAN bout us Znet. • • • • • • • • • • • • • • • • • • •
WIZnet  VIZnet  VIZnetak  OBAR(K)  FAQ  COMPACT IS  ODVIODAD  COMPACT IS  ODVIODAD	Products Technol     Products Technol     O Q&A(En)     Setial configuration     45 Setial configuration     44 How chect that Tx b     43 APt for AVR Codevis     42 heaging problem (0)     41 dynamic Simeod (0)	dogy Support	How to buy	News F News F ork solution nd smartes Inter- ips and composi- tion of the solution of the solu	CONTACT US     Contract US     Cartainers     A     Contract US     Cartainers     Contract US     Contread     Contract     Contract US     Contract     Contract US	• KOREAN About us Cnet. • • • • • • • • • • • • • • • • • • •
WIZnet  SUPPORT WIZnetAx  GAA(r)  AAA(r)  FAQ  Lead-free Status  Download  Related Link  CONTACT US	Products Technol  Products Technol  Q&A(En)    Q&A(En)	blogy Support  Support  SUBJECT  rtmp710.(0)  utfor is en(0)  ion (0)  at (0)	How to buy	News F News F ork solution and composi- pa and	CONTACT US     Partners     A	<ul> <li>KONEAN</li> <li>KONEAN</li></ul>
EVIZNEE	Products Technol     Products Technol     Products Settle configuration     As Settle configuration     As Settle configuration     As Area AVR Codevis     Anaging problem (0)     Ad dynamic timeoid (0)     do      Cell end of the anaging problem (0)     Ad dynamic timeoid (0)     do      Cell end of the anaging problem (0)     Ad dynamic timeoid (0)     do      Cell end of the anaging problem (0)     Add dynamic timeoid (0)     do      Cell end of the anaging problem (0)     Add dynamic timeoid (0)     do      Cell end of the anaging problem (0)     Add dynamic timeoid (0)		How to buy	News F News F ork solution and sonarresh hat part and component in NAME JL Vityusha Lule Dg Biter June Name	CONTACT US     CONTACT US     Cartainers     A     Contract US     Contrend     Contract     Contract     Contract US     Contract US	<ul> <li>KONEAN</li> <li>bbout us</li> <li>2net.</li> <li>2net.</li> <li>4</li> <li>4</li> <li>89</li> <li>1114</li> <li>84</li> <li>83</li> <li>103</li> <li>97</li> </ul>
EVIZNET VIZNET VIZNETAN: • GAA(F) • GAA(F) • FAQ • Lead-free Status • Download • Related Link CONTACT US Marketing & Suiss Biologicarization of Marketing & Suiss Biologicarization of Technical Support	Products Technol  Products Technol  Products Technol  Products Serial configuration  A How chect that Tx b  A How	Jogy         Support           Subject         With the provintion of the provintintion o	How to buy	News F News F ork solution and sonarresh hate part of component in NAME JL Vityusha Lule Dg Biter June Mercin June	CONTACT US     CONTACT US     Cartainers     A     Contract US     Cartainers     Contract US     Contreat     Contract     Contract US     Contract     Contract US	<ul> <li>KOREAN</li> <li>bout us</li> <li>cnet.</li> <li>cnet.</li></ul>
EVIZNET WIZNE	Products Technol     Products Technol     O Q&A(En)     O Q&A(En)     As is a AVR Codevit     As is a AVR Codevit     Anaging problem (0)     Add grant time     Simplify and the average of the ave	support           Support           With expression           SUBJECT           stang memory           subject           stang memory           subject           stang memory           subject           stang memory           subject           subject           stang memory           subject           subject <td>How to buy</td> <td>News F News F ork solution ind smartel fat part of composi- part of compos</td> <td>CONTACT US     Contract US     Contract US     Contract     Contreat     Contreat     Contract     Contract     Contract     Contr</td> <td><ul> <li>KOREAN</li> <li>About us</li> <li>Cnet.</li> <li>Cnet.</li></ul></td>	How to buy	News F News F ork solution ind smartel fat part of composi- part of compos	CONTACT US     Contract US     Contract US     Contract     Contreat     Contreat     Contract     Contract     Contract     Contr	<ul> <li>KOREAN</li> <li>About us</li> <li>Cnet.</li> <li>Cnet.</li></ul>
EVIZNEE VIZNEE VIZNEE VIZNEE O GAA(K) O GAA(K) O GAA(K) O GAA(K) O Download O Related Link CONTACT US Marketing & Sales Belesideviznee Cook Technical Support	Products Technol     Construction	SUBJECT SUBJECT SUBJECT start store subject start store subject start store subject start store	How to buy	News A News News News News News News News News	CONTACT US     Contract     Contract US     Contract     Contract US     Contract     Con	<ul> <li>KOREAN</li> <li>About us</li> <li>About us</li> <li>Cret.</li> <li>Cret.</li></ul>
EVIZNEE VIZNEE VIZNEE VIZNEE O GAA(Kr) O GAA(Kr)	Products Technol     Construction	Alegy Support	How to buy the best netw the best netw device the simplest device TCP/IPch	News F News F ork solution of solution of NAME J Utyuzha Luie Dg Biter June March June Diter June	CONTACT US     Contract     Contract US     Contract     Contract US     Contract US	<ul> <li>KOREAN</li> <li>About us</li> <li>About us</li> <li>Cret.</li> <li>Cret.</li></ul>

# COPYRIGHT NOTICE



Copyright 2006 WIZnet, Inc. All Rights Reserved.

Technical Support: <u>support@wiznet.co.kr</u> Sales & Distribution: <u>sales@wiznet.co.kr</u> General Information: <u>info@wiznet.co.kr</u>

For more information, visit our website at http://www.wiznet.co.kr



# Table of Contents

1. Overview		v		1
	1.1. Pa		kage	1
	1.2.	Fea	ture	2
	1.2.1	1.	H/W Feature	2
	1.2.2	2.	F/W Feature	2
2.	Getting S	Starte	əd	3
	2.1.	PC	Programs Install	3
	2.1.1	1.	Development Program Install	3
	2.1.2	2.	EVB B/D Test PC Program Install	3
	2.2.	Qui	ck Start	4
	2.3.	EVE	B B/D Test	5
	2.3.1	1.	Manage Program	6
	2.3.2	2.	EVB B/D Test Applications	12
	2.4.	Trou	ubleshooting Guide	16
	2.4.1	1.	Ping	16
	2.4.2	2.	Misc	16
3.	Program	mer's	s Guide	17
	3.1.	Mer	mory Map	17
	3.1.1	1.	Code & Data Memory Map	17
	3.1.2	2.	SERIAL EEPROM MAP	18
	3.2.	EVE	3 B/D Firmware	24
	3.2.1	1.	Sources	25
	3.2.2	2.	How to Compile	26
	3.2.3	3.	How to download	26
	3.2.4	4.	EVB B/D's main()	26
	3.2.5	5.	Manage Program	30
	3.2.6	6.	Applications	46
4.	Hardwar	e De	signer's Guide	90
	4.1.	Bloc	ck Diagram	90
	4.2.	Bloc	ck Description	91
	4.2.1	1.	PM-PIC24	91
	4.2.2	2.	NM7010B <sup>+</sup> MODULE	95
	4.2.3	3.	LCD	95



4	.2.4.	SRAM	96
4	.2.5.	RS232 Port	96
4	.2.6.	Expanded Board Interface	96
4	.2.7.	ICD2 Port	98
4	.2.8.	Power Regulator	98
4	.2.9.	3.3V Power On System Reset	98
4.3.	Sch	ematic	
4	.3.1.	MB-EVB-PIC	99
4	.3.2.	PM-PIC24	
4	.3.3.	NM7010B <sup>+</sup>	99
4.4.	Part	s List	100
4	1.4.1.	MB-EVB-PIC Parts List	
4	4.4.2.	PM-PIC24 Parts List	100
4	.4.3.	NM7010B <sup>+</sup>	100
4.5.	Phys	sical Specification	101
4	1.5.1.	Power Consumption	



# Figures

<fig 2.1="" :="" b="" d="" display="" evb="" lcd="" text="">4</fig>	
<fig 2.2:="" of="" output="" program="" terminal=""></fig>	
<fig 2.3:="" b="" d="" evb="" ping="" reply="" test=""></fig>	
<fig 2.4:manage="" execution="" program=""></fig>	,
<fig 2.5:="" config="" network="">7</fig>	,
<fig 2.6:="" address="" example="" ip="" setup="" source=""></fig>	
<fig 2.7:="" address="" example="" mac="" setup=""></fig>	
<fig 2.8:="" channel="" config="" menu="" of="">9</fig>	)
<fig 2.9:="" application="" client="" example="" loopback="" setting="" tcp=""></fig>	)
<fig 2.10:="" application="" of="" ping="" usage="">11</fig>	
<fig 2.11:="" application="" ping="" test="">11</fig>	
<fig 2.12:="" client="" dhcp="" test=""></fig>	,
<fig 2.13:="" loopback="" server="" tcp="" test=""></fig>	
<fig 2.14:="" client="" loopback="" tcp=""></fig>	
<fig 2.15:="" loopback="" test="" udp="">14</fig>	
<fig 2.16:="" server="" test="" web=""></fig>	
<fig 2.17:="" b="" d="" default="" evb="" of="" page="" web="">15</fig>	,
<fig 2.18:="" b="" control="" d="" evb="" of="" page="" web=""></fig>	,
<fig 3.1:="" b="" d="" evb="" map="" memory=""></fig>	,
<fig 3.2:="" eeprom="" map="" serial="">18</fig>	,
<fig 3.3:="" b="" d's="" evb="" main()=""></fig>	)
<fig 3.4:="" check_manage()=""></fig>	)
<fig 3.5:="" manage_config()=""></fig>	
<fig 3.6:="" manage_network()=""></fig>	
<fig 3.7:="" manage_channel()=""></fig>	,
<fig 3.8:="" ping_request()=""></fig>	,
<fig -="" 3.9:="" continue="" ping_request()=""></fig>	,
<fig 3.10:="" icmp="" message="" ping="" vs=""></fig>	)
<fig 3.11:="" ping()=""></fig>	,
<fig 3.12:="" displaypingstatistics()=""></fig>	
<fig 3.13:="" sendpingreply()="">44</fig>	
< FIG 3.14 : LOOPBACK_TCPS() >	; )
<fig 3.15:="" loopback_tcpc()="">49</fig>	)
<fig 3.16:="" loopback_udp()=""></fig>	)



<fig 3.17:="" flow="" http="" message=""></fig>	52
<fig 3.18:="" web_server()=""></fig>	55
<fig 3.19:="" proc_http()=""></fig>	56
<fig 3.20:="" parse_http_request()=""></fig>	58
<fig 3.21:="" find_http_uri_type()=""></fig>	59
<fig &="" 3.22:="" get_http_parse_value()="" get_http_uri_name()=""></fig>	59
<fig 3.23:="" netconf.cgi="" processing=""></fig>	60
<fig 3.24:="" lcdnled.cgi="" processing=""></fig>	61
<fig 3.25:="" dhcp="" flow="" message=""></fig>	63
<fig 3.26:="" dhcp="" format="" message=""></fig>	64
<fig 3.27:="" dhcp="" field="" format="" message's="" option=""></fig>	65
<fig 3.28:="" init_dhcp_client()=""></fig>	67
<fig 3.29:="" getip_dhcps()=""></fig>	68
<fig 3.30:="" by="" client="" dhcp="" flow="" message="" state=""></fig>	70
<fig 3.31:="" check_dhcp_state()=""></fig>	71
<fig &="" 3.32:="" check_dhcp_timeout()="" parse_dhcpmsg()=""></fig>	72
<fig &="" 3.33:="" dns="" domain="" flow="" message="" name="" structure="" system=""></fig>	74
<fig 3.34:="" dns="" format="" message=""></fig>	75
<fig 3.35:="" format="" header="" section=""></fig>	75
<fig 3.36:="" format="" question="" section=""></fig>	75
<fig 3.37:="" format="" recode="" resources=""></fig>	76
<fig &="" 3.38:="" gethostbyaddr()="" gethostbyname()=""></fig>	78
<fig 3.39:="" dns_query()=""></fig>	79
<fig 3.40:="" dns_makequery()=""></fig>	80
<fig 3.41:="" example="" field="" of="" qname="" question="" section="" transformation=""></fig>	81
<fig 3.42:="" dns_parse_response()=""></fig>	83
<fig &="" 3.43:="" dns_answer()="" dns_parse_question()=""></fig>	85
<fig 3.44:="" parse_name()=""></fig>	86
<fig 3.45:="" compression="" dns="" message="" scheme=""></fig>	87
<fig 4.1:="" b="" block="" d="" diagram="" evb=""></fig>	90
<fig 4.2:="" dimension="" module="" pm-pic24=""></fig>	91



# Tables

<table 1-1:="" b="" contained="" d="" evb="" in="" items="" list="" of="" the=""></table>	.1
<table 1-2="" :="" cd="" contents="" of="" software=""></table>	.1
<table 2-1="" :="" properties="" setting="" terminal=""></table>	.4
<table 2-2="" :="" b="" d="" default="" evb="" information="" network=""></table>	6
<table 2-3="" :="" config="" menu="" network="" of=""></table>	.7
<table 2-4="" :="" b="" channel="" d="" default="" evb="" information=""></table>	.8
<table 2-5="" :="" channel="" config="" menu="" of=""></table>	.9
<table 2-6="" :="" application="" channel="" iinchip™="" type=""></table>	.9
< TABLE 2-7 APPLICATION DEFAULT VALUE >	10
<table 3-1:="" definition="" device="" map=""></table>	18
<table 3-2:="" definition="" eeprom="" map="" serial=""></table>	19
<table 3-3:="" information="" system=""></table>	20
<table 3-4:="" data="" definition="" sysinfo="" type=""></table>	20
<table 3-5:="" access="" functions="" information="" system=""></table>	20
<table 3-6:="" information="" network=""></table>	21
<table 3-7:="" data="" definition="" netconf="" type=""></table>	21
<table 3-8:="" access="" functions="" information="" network=""></table>	21
<table 3-9:="" channel="" information=""></table>	22
<table 3-10:="" application="" channel="" type=""></table>	22
<table 3-11:="" chconf="" data="" definition="" type=""></table>	23
<table 3-12:="" access="" channel="" function="" information=""></table>	23
<table 3-13:="" b="" d="" evb="" sources=""></table>	25
< TABLE 3-14 : IINCHIP™'S DEFINE OPTION >	26
<table 3-15:="" b="" d's="" evb="" functions="" in="" main()="" reference=""></table>	28
<table 3-16:="" at="" caller="" function="" manage="" program=""></table>	32
<table 3-17:="" functions="" in="" manage_config()="" reference=""></table>	34
<table 3-18:="" application="" by="" constraint="" types=""></table>	35
< TABLE 3-19: REFERENCE FUNCTIONS IN MANAGE_CHANNEL() >	36
<table 3-20:="" data="" definition="" pingmsg="" type=""></table>	40
<table 3-21:="" data="" definition="" pinglog="" type=""></table>	40
<table 3-22:="" functions="" in="" ping_request()="" reference=""></table>	45
<table 3-23:="" functions="" in="" loopback_tcps()="" reference=""></table>	47
<table 3-24:="" functions="" in="" loopback_tcpc()="" reference=""></table>	49
<table 3-25:="" functions="" in="" loopback_udp()="" reference=""></table>	51



<table 3-26:="" browser's="" http="" operation="" procedure="" request="" web=""></table>	2
<table 3-27:="" format="" http="" message=""></table>	3
<table 3-28:="" and="" b="" between="" browser="" d="" evb="" http="" message="" web=""></table>	ł
<table "evbctrl.html"="" 3-29:="" at="" environment="" system="" usage="" variables=""></table>	7
<table "st_http_request"="" 3-30:="" data=""></table>	3
<table 3-31:="" functions="" in="" reference="" web_server()=""></table>	2
<table 3-32:="" data="" dhcp="" message="" type="">64</table>	ł
<table 3-33:="" code="" definition="" dhcp="" message="" option=""></table>	5
<table &="" 3-34:="" client="" definition="" dhcp="" state="" timeout=""></table>	)
<table 3-35:="" dhcp="" field="" flag="" message="" setup="">70</table>	)
<table 3-36:="" client="" dhcp="" functions="" in="" reference=""></table>	3
<table 3-37:="" data="" dns="" message="" type="">77</table>	7
<table 3-38:="" at="" definition="" dns_query()="" query="" type="">77</table>	7
<table 3-39:="" and="" constants="" header="" in="" macro="" section="" used=""></table>	ł
<table &="" 3-40="" :="" at="" constants="" definition="" field="" qclass="" qtype=""></table>	2
<table 3-41="" :="" at="" constant="" definition="" field="" header="" rcode="" section's=""></table>	ł
<table 4-1:="" description="" module="" pin="" pm-pic24=""></table>	2
<table 4-2:="" description="" isp="" pin=""></table>	ł
<table 4-3:="" description="" lcd="" pin=""></table>	5
<table 4-4:="" board="" description="" expanded="" interface="" pin=""></table>	5
< TABLE 4-5 EVB B/D POWER CONSUMPTION >	i



# 1. Overview

EVB-PIC24 (referred to as "EVB B/D or EVB" from here on) is iinChip™ Evaluation B/D for PIC developer.

# 1.1.Package

When purchasing EVB B/D please make sure you have all the following contents.

	ltem	Quantity
	MB-EVB-PIC EVB Main Board	1
EVB B/D	PM-PIC24 MCU Module (Plugged In MB-EVB- PIC)	1
	NM7010B <sup>+</sup> (Plugged In MB-EVB-PIC)	1
	Power Adaptor (5V)	1
		Option
A	Software CD	1
Accessory	UTP Cable	1
	Serial Cable	1

<table 1-1<="" th=""><th>List of</th><th>Items</th><th>Contained</th><th>in</th><th>the</th><th>FVR</th><th>B/D&gt;</th></table>	List of	Items	Contained	in	the	FVR	B/D>
	LISCO	nems	containeu		uie		0,0/

<	Table	1-2 :	Con	tents	of	Softwa	are	CD>

	Directo	Contents	
EVB-	DOC	Datasheet	All sorts of Datasheet
PIC24		Development	ASM, C, Library Manual
		Manual	Users Manual
HW		Schematics	All sorts of schematics
		Part List	All sorts of Part List
	SW	Firmware	EVB B/D Firmware
		PC Utility	All sorts of Tool Program
NM7010B <sup>+</sup>			
W3150A <sup>+</sup>			

• The contents of Software CD could be changed by version. Please check "ReadMe.txt" of CD.



# 1.2. Feature

1.2.1. H/W Feature

EVB B/D consists of 3 kinds B/D

- PM-PIC24
- MCU : PIC24FJ128GA006, 16MHz
- RAM : 8KB SRAM (Internal), 32KB SRAM(External)
- ROM : 128KB Flash (PIC24 Internal Flash)
- JTAG, ICSP, ICE
- MB-EVB-PIC
  - Power : 5V Adaptor
  - UART : RS-232 Serial Port, (1 Port, default 57600 Baud Rate)
  - LCD Display : 16x2 Charactor LCD
- Network Module (NM7010B<sup>+</sup>)

F/W Feature

- iinChip™ : Hardwired TCP/IP Chip W3150A<sup>+</sup>
- PHY : RTL8201CP(RealTek), 10/100 BaseT(X) Auto Negotiation
- MagJack : RD1-125BAG1A (UDE) , Integrated Transformer(1:1)

### Link & ACT LEDs

#### 1.2<mark>.2.</mark>

The F/W Feature of EVB B/D is made up of two parts.

### Manager mode

- Network Config : MAC, Source IP, G/W IP, S/N, DNS IP Setup
- Channel Config : iinChip<sup>™</sup> Test Application Setup for each Channel
- Ping Test : Ping Request Test with DNS
- Application mode
- Loopback TCP Server : TCP Server Mode Test Application
- Loopback TCP Client : TCP Client Mode Test Application
- Loopback UDP : UDP Test Application
- Web Server : Web Server Test Application
- DHCP Client : Dynamic Network Config using DHCP Server



# 2. Getting Started

# 2.1. PC Programs Install

2.1.1. Development Program Install

Please refer to Microchip(www.microchip.com) for more information.

2.1.1.1. MPLAB IDE(Integrated Development Environment)

It included following features – assembler, linker, simulator(with peripheral), VDI(Visual Device Initializer) and full featured debugger.

For installation and usage of MPLAB IDE, refer to the related manual.

Firmware of EVB B/D is currently using MPLAB IDE Version 7.40 and can be changed with newer IDE.

## 2.1.1.2. MPLAB C30

It is a full-featured ANSI compliant C compiler for the Microchip 16-bit devices. Also It is fully compatible with MPLAB IDE.

For installation and usage of MPLAB IDE, refer to the related manual.

Firmware of EVB B/D is currently using MPLAB C30 Version 2.02 and can be changed with newer compiler.

## 2.1.1.3. ROM File Maker Program

ROM File Maker Program is a utility program that provides convenience in using simple 'ROM File System' for EVB B/D. The reason that ROM File Maker Program is used in EVB B/D is to access Web Pages for Web Server Test Application as 'ROM File System'. Refer to "ROM File Maker Manual Vx.x.pdf" for further instruction on installation and ROM File Maker Program.

## 2.1.2. EVB B/D Test PC Program Install

## 2.1.2.1. Loopback Test Program (AX1) Install

Loopback Test Program (referred to as "AX1" from here on) is a program to evaluate the performance of iinChip<sup>™</sup> and does the Loopback on file and packet data in connection with EVB B/D channel applications such as Loopback TCP Server/Client and Loopback UDP. Please refer to "**AX1 Manual Vx.x.pdf**" for installation and usage.



# 2.2. Quick Start

After the confirming the Package of EVB B/D, test EVB B/D in the order shown below.

- Connect test PC to EVB B/D using UTP cross cable directly. Connect test PC to EVB B/D using serial cable directly. Connect 5V power adaptor to EVB B/D
- Confirm the network information of Test PC as the following Source IP Address : 192.168.0.3
   Gateway IP Address : 192.168.0.1
   Subnet Mask : 255.255.255.0
- ③ Install AX1 on Test PC. Refer to Chapter 2.1.2.1
- ④ After the execution of serial terminal program(like Hyperterminal), set up the properties as the following.

### <Table 2-1 : Terminal Properties Setting>

Properties	Setting Value				
Bits Per second(Baud Rate)	5 <mark>7600 b</mark> ps				
Data Bits	8 <mark>Bits</mark>				
Stop Bits	1 Bit				
Parity	No				
Flow Control	None				

After the completion of terminal setup, connect EVB B/D and wait.

5 Turn on the power switch of EVB B/D

Following items should be checked upon power on

- Check lighting on power LED(D1) of EVB B/D when powering on
- Check if LEDs of D2 and D3 blink three times by turns.
- Check if Text LCD display of EVB B/D outputs in the way shown in <Fig 2.1> and shown in <Fig 2.2> on the Terminal Program







**EVB-PIC24 User's Manual** 



#### 2.3.1. Manage Program

Manage Program is a program that is executed upon receiving character 'M' or 'm' from the terminal program within 7 seconds when doing the manual reset of EVB B/D and EVB B/D power On. This program sets up the Channel Application of iinChip<sup>™</sup> to be tested. And perform certain ping request test with DNS server.



#### <Fig 2.4:Manage Program Execution >

#### 2.3.1.1. Network Configuration

It selects Network Information that is used in EVB B/D. When choosing '1' at terminal Program of <Fig 2.7>, Network Information of EVB B/D can be set as desired. The default Network Information of EVB B/D is shown in <Table 2-2>.

Network Information	Default Value
MAC Address	00.08.DC.00.00.24
Source IP Address	192.168.0.2
Gateway IP Address	192.168.0.1
Subnet Mask	255.255.255.0
DNS Server IP Address	0.0.0.0

#### <Table 2-2 : EVB B/D Default Network Information>

If "Network Config" menu is selected on Manage Program, menu shown in <Fig 2.8> can be displayed and each function is described in <Table 2-3>.



Select ? 1 \_\_\_\_\_ NETWORK CONFIG -----D : Display config : Source IP 1 : Gateway IP 2 : Subnet Mask 3 4 : DNS Server IP M : MAC address : memory Allocation A F : Factory reset Е : Exit \_\_\_\_\_ Select ?

<Fig 2.5: Network Config>

Menu	Description
D : Display Config	Display curren <mark>t N</mark> etwork Inf <mark>ormati</mark> on
1 : Source IP Address	Sets up Source IP Address
2 : Gateway IP Address	Sets up Gateway IP Address
3 : Subnet Mask	Sets up Subnet Mask
4 : DNS Server IP	Sets up DNS Server IP Address
	<warning> DNS Server is information needed for "Ping Request"</warning>
	Test and transformation of Domain Name into IP address and so it
	must be set up as Static IP Address.
'A' or 'a'	Sets up Memory Allocation.
	iinChip™ Memory Size Register.(RMSR,TMSR)
	Refer to "W3150A <sup>+</sup> Datasheet.pdf".
F : Factory Reset	Initialization of the system with the default value.
	Refer to <table 2-2=""></table>
'M' or 'm'	Sets up MAC Address.
	<warning> This value is not change when Factory Reset.</warning>
E : Exit	Exit "Net Config"



<Fig 2.9> is an example of setting the Source IP of EVB B/D in Network Config

```
_____
NETWORK CONFIG
    _____
  : Display config
D.
  : Source IP
   Gateway IP
    Subnet Mask
   DNS Server IP
    MAC address
м
   memory Allocation
A
  : Factory reset
Е
 : Exit
            ____
Select ? 1
Source IP ? 192.168.0.100
```

#### <Fig 2.6: Source IP Address Setup Example>

<Fig 2.10> is an example of setting the MAC address of EVB B/D in Network Config

NE	Τū	JORK CONFIG
D	:	Display config
1	:	Source IP
2	:	Gateway IP
3	:	Subnet Mask
4	:	DNS Server IP
М	:	MAC address
A	:	memory Allocation
F	:	Factory reset
Е	:	Exit
Se	16	ect ? m
MA	С	Address ? 00.08.dc.00.00.20

#### <Fig 2.7: MAC address Setup Example>

2.3.1.2. Channel Config

It set up Test Application that can be operated in iinChip<sup>™</sup> 4 channel of EVB B/D. If '2' is selected, each channel can be set up. The default iinChip<sup>™</sup> Channel Information is shown in <Table 2-4>.

iinChip™ Channel	Test Application
1 <sup>st</sup>	Loopback TCP Server (Port 5000)
2 <sup>nd</sup>	Loopback TCP Server (Port 5000)
3 <sup>rd</sup>	Loopback TCP Server (Port 5000)
4 <sup>th</sup>	Loopback TCP Server (Port 5000)

#### <Table 2-4 : EVB B/D Default Channel Information>



If "Channel Config" menu is selected in manage program, <Fig 2.11> is displayed and the functionality of each menu is same as <Table 2-5>.

1																		
S	e	1	e	С	t		?		2									
-	-	_	-	-	-	-	-	-	-	-	-	-	-	-	_	-	-	
С	H	A	Ν	N	E	L		С	0	Ν	F	Ι	G					
-	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	-	
ю		:		D	i	3	p	1	a	У		С	o	n	f	i	g	
1		:		1	3	t		С	h	a	n	n	e	1				
2		:		2	n	d		С	h	a	n	n	e	1				
3		:		3	r	d		С	h	a	n	n	e	1				
4		:		4	t	h		С	h	a	n	n	e	1				
F		:		F	a	с	t	o	r	у		R	e	3	e	t		
E		:		E	x	i	t											
-	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	
s	e	ı	e	c	t		2											

<Fig 2.8: Menu of Channel Config>

Menu	Description
D : Display Config	Output of current <mark>Set</mark> up Tes <mark>t Appli</mark> cation Type b <mark>y eac</mark> h iinChip™
	Channel
0 : 1 <sup>st</sup> Channel	Test Application Type Setup at iinChip™ No. "0" Channel
	<warning> As developing EVB B/D, DHCP Client Application Setup is</warning>
	possible at only No. "0" Channel.
1 : 2 <sup>nd</sup> Channel	Test Application Type Setup at iinChip™ No. "1" Channel
2 : 3 <sup>rd</sup> Channel	Test Application Type Setup at iinChip™ No. "2" Channel
3 : 4 <sup>th</sup> Channel	Test Application Type Setup at iinChip™ No. "3" Channel
F : Factory Reset	Initialization into original Setup status. Refer to <table 2-4=""></table>
E : Exit	Exit "Channel Config"

Available Setup Test Application by each iinChip™ Channel is shown as <Table 2-6>

#### <Table 2-6 : iinChip™ Channel Application Type>

Application Type	Description
No Use	Not used
DHCP Client	Receiving Network Information of EVB B/D from DHCP Server
	dynamically
	<warning> If DHCP Server is not existing in LAN it sets back to</warning>
	default value after certain amount of time
TCP Loopback Server	TCP Server Test Program
	<warning> EVB B/D : TCP Server, AX1 : TCP Client</warning>



TCP Loopback Client	TCP Client Test Program
	<warning> EVB B/D : TCP Client, AX1 : TCP Server</warning>
Loopback UDP	UDP Test Program
Web Server	Web Server Test Program

Other application types except for "DHCP Client" can be repeatedly set up regardless of channel. <Fig 2.12> shows an example of setting the 2<sup>nd</sup> Channel of iinChip™ as "TCP Loopback Client"

When inputting simply [ENTER] without IP address or port number, the default value is applied. <Table 2-7> shows default values required for each application.

#### <Fig 2.9: Loopback TCP Client Application Setting Example>

Application Type	Default Value
DHCP Client	None
TCP Loo <mark>pback</mark> Server	Listen Port Number : 5000
TCP Loopback Client	Server IP Address : 192.168.0.3
	Server Port Number : 3000
Loopback UDP	Source Port Number : 3000
Web Server	HTTP Port Number : 80

#### < Table 2-7 Application Default Value >



#### 2.3.1.3. Ping Application Test

Ping Application Test is a program created for IP RAW Channel Evaluation of iinChip<sup>™</sup> and sends Ping request to certain peer and receives Ping Reply. This program is set up identically with the ping command in the DOS prompt. It's executed when '3' is chosen <Fig 2.4:Manage Program Execution >

🖊 Mini Term		_1012
File (E) Configure (C) T	ransfer (_) Help (H)	
z, z		
2 : Channel Config 3 : Ping Ann Test	1	
F : Factory Reset		
E : Exit		
	-	
Select ? 3		
Ping Reqeust progr	am started	
Usage : ping [-t]	[-a] [-n count] [-1 size] [-w timeout] destination-list	st
Option :		
-t	Ping the specified host until stopped.	
	To see statistics and continue - type Control-Break;	
	To stop - type Control-C.	
-a	Resolve addresses to hostnames	
-n count	Number of echo requests to send.	
-1 5126	Send Buller Size.	
-w timeout	Timeout in milliseconds to wait for each reply.	
DINCS		
FING2		
Connected 57600	8-None-1 None Canture : OFF	O PY O TY

<Fig 2.10: Usage of Ping App<mark>licatio</mark>n >

<Fig 2.13> displays the execution screen of Ping Application and shows how to use the Ping Application.

<Fig 2.14> shows the real example of sending the Ping Request to the destination and receiving the Ping Reply.

Mini Term	>
Pried Collinguie C Hairsler () Help ()	
-n count Number of echo requests to send.	
-1 Size Send Buller Size.	
-w timeout Timeout in milliseconds to wait for each reply.	
PING> ning www.wahoo.co.kr	
DNS SERVER:164.124.101.2	
Ping Request to 202.43.214.151[www.yahoo.co.kr]	
Pinging 202.43.214.151 with 32 bytes of data :	
Reply from 202.43.214.151 : bytes=32, time<=2ms	
Reply from 202.43.214.151 : bytes=32, time<=2ms	
Reply from 202.43.214.151 : bytes=32, time<=2ms	
Reply from 202.43.214.151 : bytes=32, time<=2ms	
Ding statistics .	
Packets: Sent = 4. Received = 4. Lost = 0	
Ping Reply Packets = 4	
13. URB/2006 - 2008/00/-2006 - 2006/00/00/00/00/00/00/00/00/00/00/00/00/	
D TVO	
PING	
Connected E7000 9 Marco 1 Marco Contrus : OEE	O DY O TY
Connected 57000, 6-NORE-1, NORE Capture : OFF	W KX W IX

To terminate the Ping Application type, "exit" in "PING>" prompt.



2.3.2. EVB B/D Test Applications

2.3.2.1. DHCP Client

DHCP Client Application is an application that dynamically assigns network information for EVB B/D from DHCP Server. To test DHCP Client, first of all, iinChip<sup>™</sup> 1<sup>st</sup> channel application type must set up as "DHCP Client" using [Manager>>Channel Config>>0th Channel] Menu.

Refer to Chapter 2.4.1.2

<Fig 2.15> is the screen that DHCP Client successfully obtains network information. Note that DHCP Client will be set with default network information if DHCP Server does not exist or is not able to obtain network information from DHCP Server.

Mini Term	
File (E) Configure (C) Transfer (I) Help (H)	
£s ₽x 2	
: DHCH Client Start.	~
<pre>et network information from DHCP Server ##################################</pre>	
/W Version : 2.0.0.1 /W Version : TB-B1 AC Addr : 00.08 DC 00.00 00 ource IP : 210.221.197.33 ateway IP : 210.221.197.33 ubnet Mask : 255.255.255.0 NS Server IP : 164.124.101.2 em alloc : 55	
######################################	~

<Fig 2.12: DHCP Client Test>

### 2.3.2.2. Loopback TCP Server

Loopback TCP Server Application is an application that loops back any file or packet data through TCP Channel connected with "AX1" Program of Test PC. First of all, set any channel as "Loopback TCP Server" Application Type using [Manager>>Channel Config] menu of EVB B/D to test Loopback TCP Server. When setting up "Loopback TCP Server" Application Type of EVB B/D, you can set listen port Value to any value. Here, it's set as the default value, 5000. Refer to <u>Chapter 2.4.1.2</u>



After the setup of EVB B/D is complete, run "AX1" at Test PC then try the connection to the IP Address. When the connection between EVB B/D and "AX1" is successful, loop back the data. Refer to "**AX1 Manual Vx.x.pdf**"

So	uı	cce	I	P		:	1	92	2.	1	68		ο.	. 2										
Gε	ate	eway		IP		:	1	92	2.	1	68		ο.	. 1										
ຮະ	ıbı	net	M	as	k	:	2.	55	5.	2	55		25	55		0								
DN	IS	Ser	v	er	IP	:	0	. c	э.	0	. 0													
M A	C	Add	r	:	0x	00.	0:	хC	18		0 x	D	с.	0	х	0(	٥.	0:	κO	ο.	. 0	x3	5	
##	###	####	#	##	###	###	##	##	##	#	##	#	##	##	#	#1	##	#1	##	##	¥#	##	##	###
0	:	Loo	p	- B	ack	T	P	02	ŏe	r	ve	r		št	a	r۱	ce	d						
1	:	Loo	p	- B	ack	T	P	2	ŏe	r	ve	r	2	št	a	rt	ce	d						
2	:	Loo	p	- B	ack	TO	P	92	ŏe	r	ve	r	2	št	a	rt	te	d						
3	:	Loo	p	- B	ack	TO	P	2	še	r	ve	r	2	št	a	r۱	ce	d						
0	:	Con	n	ec	ted	by	7	19	92	•	16	8	. 0	).	3	0	(2	3 :	13	)				
	Pee	er Co	nr	nec	tion	Info	orm	na	tic	n	in	0	C	ha	Inr	ne	10	ofi	iin(	Ch	nip	TM		

<Fig 2.13: Loopback TCP Server Test>

2.3.2.3. Loopback TCP Client

Loopback TCP Client Application is an application that loop backs any file and packet data through TCP channel connected with "AX1" Program of Test PC

After running the "AX1" on the server, set any channel of iinChip<sup>™</sup> as "Loopback TCP Client" Application type using [Manager>>Channel Config] menu of EVB B/D.

When setting up the "Loopback TCP Client" Application type of EVB B/D, set the Server IP as the IP Address of the Test PC and set Server Port as the waiting Server Port Number(3000). Refer to <u>Chapter 2.4.1.2</u>. After setting up EVB B/D is complete, exit from the manager program and run EVB Test Application. If EVB B/D is connected to "AX1" successfully, loop back the desired data. Refer to "**AX1 Manual Vx.x.pdf**"

So	our	ce IP	:	192.168	.0.2		
Ge	ate	eway IP	:	192.168	.0.1		
SI	ıbr	net Mask	:	255.255	.255.	0	
DI	IS	Server IP	:	0.0.0.0			
MJ	AC	Addr : Ox0	00.	.0x08.0x	DC.Ox	00.0x00.	0x35
##	###	********	;#;	*#######	#####	########	*#######
0	:	Loop-Back	Т	CP Serve	r Sta	irted.	
1	:	Loop-Back	T	CP Clien	t Sta	irted.	
Z	:	ьоор-васк	10	JP Serve	r sta	irtea.	
3	:	Loop-Back	T	CP Serve	r Sta	irted.	
1	:	Connected	by	y 192.16	8.0.3	0(2827)	Peer Connection Information
					_		in 1 channel of iinChip™

<Fig 2.14: Loopback TCP Client>

### 2.3.2.4. Loopback UDP

Loopback UDP Application is an application that loops back any File and Packet Data through UDP Channel connected with "AX1" Program of Test PC. First of all, to test Loopback UDP, set up any Channel of iinChip<sup>™</sup> as "Loopback UDP" Application Type using [Manager>>Channel Config] Menu of EVB B/D.

In setting up "Loopback UDP" Application type, set Source Port as any value. Here, it's set with 3000. Refer to <u>Chapter 2.4.1.2</u>



After EVB B/D setup is over, loop back desired data with IP Address and UDP Source Port of EVB B/D using menu or Icon related to UDP.

Refer to "AX1 Manual Vx.x.pdf".

#### <Fig 2.15: Loopback UDP Test>

2.3.2.5. Web Server

Web Server Application sends and receives web pages and EVB B/D control data etc. through HTTP Channel connected with web browser. For Web Server testing, set up any channel of iinChip™ as "Web Server" Application Type using [Manager>>Channel Config] Menu of EVB B/D.

When setting up "Web Server" Application Type of EVB B/D, set HTTP port as any value. Here, it's set to 80, the default value. Refer to <u>Chapter 2.4.1.2</u>.

After setup for EVB B/D, run Web browser in the Test PC, type the URL(<u>http://192.168.0.2/</u>) of the EVB B/D in the address field and connect to EVB B/D.

```
Source IP
             : 192.168.0.2
Gateway IP
            : 192.168.0.1
             : 255.255.255.0
Subnet Mask
DNS Server IP : 0.0.0.0
MAC Addr : 0x00.0x08.0xDC.0x00.0x00.(
0 : Loop-Back TCP Server Started.
1
 : Loop-Back TCP Client Started.
   Loop-Back UDP Started.
  :
    Web Server Started.

    Web Server Application Log and

  : Connected by 192.168.0.30(2313)
                                     Peer Connection Information
```

#### <Fig 2.16: Web Server Test>

If the web browser successfully connected to HTTP port of EVB B/D, the Web Page of <Fig 2.20> can be viewed. In case Web Page of <Fig 2.20> is not shown, refresh the screen using the "Refresh" function of the web browser.





<Fig 2.17: Default Web Page of EVB B/D>

If [Control] button on the Web Page in <Fig 2.20> is clicked, it can set the network information or show the web page that can turn on/off LEDs(D1,D2) and display rows of text on Text LCD display.

Source IP       192.168.000.002         Gateway IP       192.168.000.001         Subnet Mask       255.255.255.000         DNS Server IP       000.000.000.000         MAC Address       00,08,DC,00,00,35         Network Config         192.168.000.002       □         LED 0       ♀         LED 1       ♦         LCD & LED Config       □		WIZnet
Gateway IP       192.168.000.001         Subnet Mask       255.255.255.000         DNS Server IP       000.000.000         MAC Address       00,08,DC,00,00,35         Network Config         192.168.000.002       □         LED 0       ♀         LCD & LED Config       ↓	Source IP	192.168.000.002
Subnet Mask       255.255.255.000         DNS Server IP       000.000.000         MAC Address       00,08,DC,00,00,35         Network Config         192.168.000.002       □         LCD & LED Config	Gateway IP	192.168.000.001
DNS Server IP       000.000.000         MAC Address       00,08,DC,00,00,35         Network Config         192.168.000.002         □       LED 0         ②         LCD & LED Config	Subnet Mask	255.255.255.000
MAC Address 00,08, DC, 00, 00, 35 Network Config	DNS Server IP	000.000.000.000
Network Config 192.168.000.002 □ LED 0 ♀ LED 1 ♀ LCD & LED Config	MAC Address	00, 08, DC, 00, 00, 35
	[192.168.000.	002 □ LED 0 ♀ LED 1 ☆ LCD & LED Config
<fig 2.18:="" b="" control="" d="" evb="" of="" page="" web=""></fig>	<fig 2.18<="" th=""><td>: Web Page of EVB B/D Control&gt;</td></fig>	: Web Page of EVB B/D Control>



# 2.4. Troubleshooting Guide

### 2.4.1. Ping

When you can not reach EVB B/D by Ping command,

- Step 1. Did you connect correctly between test PC and EVB B/D with UTP cable?
- Step 2. Did you change your test PC's network environment (IP address, Gateway,

Subnet)? If no, you should change it first as follows:

IP address: 192.168.0.3

Gateway address: 192.168.0.1

Subnet Mask: 255.255.255.0

Step 3. Whether NM7010B<sup>+</sup>'s Link LED(left LED from rear view) is on?

If it is off, you'd better check whether the UTP cable works correctly.

### 2.4.2. Misc.

When the serial terminal screen remains blank with the power on after a connection is made

- Step 1. Check the connection condition of the serial cable.
- Step 2. Check if the COM Port numbers of the PC and terminal coincide.
- Step 3. Check the terminal's baud rate 57600.



# 3. Programmer's Guide

# 3.1. Memory Map

## 3.1.1. Code & Data Memory Map

Memory Map of EVB-PIC24 has Program Memory 128 Kbytes, SRAM 8Kbytes. External Space(Parallel Master Port) divided into SRAM Area and W3150A<sup>+</sup> Area. Other than these, there is 4Kbytes Serial EEPROM. Various kinds of environmental variables are recorded on this EEPROM.

<Fig 3.1>, <Table 3-1> are representations of System Memory Map of EVB B/D.





<table 3-1:="" definition="" device="" map=""></table>													
Devi	ice	Map Define										Source Co	ode
iinChi	ip™	#defin	eDEF_	IINC	HIP_N	/AP_BAS	Ξ	0x8000			iinCh	ip/iinchip	_conf.h
		#if (	DEF_IINC	HIP	_BUS_	==D	EF_	IINCHIP_DIRECT_	MODE	)			
		#defi	ne COMM	10N_	BASE	DEF_I	NC	HIP_MAP_BASE	-				
		#else											
		#defi	ne COMM	10N_	BASE	0x0000							
		#endif											
		#define DEF IINCHIP MAP TXBUF (COMMON BASE + 0x4000)											
	#define DEF IINCHIP MAP RXBUF (COMMON BASE + 0x6000)												
3.1.2.       SERIAL EEPROM MAP <fig 3.2="">, <table 3.2=""> are representations of Serial EEPROM의 Map.         Refer to "evb/config.h" and "evb/config.c."         Addr 0 1 2 3 4 5 6 7 8 9 A B C D E F         0x00       SYS_TEST         SYS_VER       AUTO         BESET</table></fig>								F					
0x20	NET_'	TEST		NE	ET_MA	IC		NET_S	IP			NET_GWI	P
0x30		NE'	Γ_SN		1	VET_DNS		NET_MEMALLOC					
0x50	СН_Т	EST	CH_ TYPE_0	CH POR	H_ (T_0	C	снл	DEST_IPO	CH_ TYPE_1	CH POR	H_ T_1	CH_DEST	LIP1
0x60	C TY	CH_ PE_2	CH_ PORT.	_2	СН	LDEST_IP:	2	CH_ TYPE_3	CH_ PORT_3	C	CH_DEST_IP3		

<Fig 3.2: Serial EEPROM Map>



	#define SYS_INFO	0x00
Custom	#define SYS_TEST	(SYS_INFO)
System	#define SYS_VER	(SYS_TEST + 2)
Information	#define SYS_AUTORESET	(SYS_VER + 4)
	#define SYS_ANY_PORT	(SYS_AUTORESET + 1)
	#define NET_CONF	0x20
	#define NET_TEST	(NET_CONF)
	#define NET_MAC	(NET_TEST+2)
Network	#define NET_SIP	(NET_MAC + 6)
Information	#define NET_GWIP	(NET_SIP + 4)
	#define NET_SN	(NET_GWIP + 4)
	#define NET_DNS	(NET_SN + 4)
	#define NET_MEMALLOC	(NET_DNS + 4)
	#define CH_CONF	0x50
	#define CH_TEST	(CH_CONF)
	#define CH_TYPE_0	(CH_TEST + 2)
	#de <mark>fine CH_</mark> PORT_ <mark>0</mark>	(CH_TYPE_0 + 1)
	#de <mark>fine CH_DESTIP_</mark> 0	(CH_PORT_0 + 2)
	#define CH_TYPE_1	(CH_DESTIP_0 + 4)
Channel	#define CH_PORT_1	(CH_TYPE_1 + 1)
Information	#define CH_DESTIP_1	(CH_PORT_1 + 2)
	#define CH_TYPE_2	(CH_DESTIP_1 + 4)
	#define CH_PORT_2	(CH_TYPE_2 + 1)
	#define CH_DESTIP_2	(CH_PORT_2 + 2)
	#define CH_TYPE_3	(CH_DESTIP_2 + 4)
	#define CH_PORT_3	(CH_TYPE_3 + 1)
	#define CH_DESTIP_3	(CH_PORT_3 + 2)

#### <Table 3-2: SERIAL EEPROM MAP Definition>



### 3.1.2.1. System Information

System Information area is used in recording System Information such as Firmware Version of EVB B/D.

Name	Description	Default Value				
SYS_TEST	System Information's Valid Check	0xA5A5 – Valid				
		Others – Invalid				
SYS_VER	F/W Version	0xAABBCCDD (AA.BB.CC.DD)				
SYS_AUTORESET	If you set any environmental variable	0x01 – System Auto Reset				
	up, Auto Reset Check	Others – No Reset				
SYS_ANY_PORT	Using Any Port Number at Socket	1000 ~ 65535				
	creation					

#### <Table 3-3: System Information>

System Information is accessed as SYSINFO Data Type.

Type Definition	Instance	
typedef struct _SYSINFO	SYSINFO <b>SysInfo</b> ;	
u_int test; u_long ver;		
u_char auto_reset;		
u_int any_port;		
}SYSINFO;		

## <Table 3-4: SYSINFO Data Type Definition>

#### <Table 3-5: System Information Access Functions>

Function	Description
void set_sysinfo(SYSINFO* pSysInfo)	Save the System Information
void get_sysinfo(SYSINFO* pSysInfo)	Get the System Information



#### 3.1.2.2. Network Information

Network Information is used in recording Network Configuration information to be used for EVB B/D.

Name	Description	Default Value				
NET_TEST	Network Information Valid Check	0xA5A5 – Valid				
		Others – Invalid				
NET_SIP	Source IP Address	0xC0 A80002 (192.168.0.2)				
NET_GWIP	Gateway IP Address	0xC0 A80001 (192.168.0.1)				
NET_SN	Subnet Mask	0xFFFFF60 (255.255.255.0)				
NET_DNS	DNS Server IP Address	0x00000000 (0.0.0.0)				
NET_MEMALLOC	iinChip™ Memory Allocation	0x55				

#### <Table 3-6: Network Information>

Network Information is accessed as NETCONF Data Type.

Type Definition	Global Instance
typedef struct _NETCONF {	NETCONF NetConf;
u_int test; u_char mac[6]; u_long sip; u_long gwip; u_long sn; u_long dns; u_char mem_alloc;	
}NETCONF;	

## Table 3-7: NETCONF Data Type Definition>

#### <Table 3-8: Network Information Access Functions>

Function	Description
<pre>void set_netconf(NETCONF* pNetConf)</pre>	Save the Network Information
<pre>void get_netconf(NETCONF* pNetConf)</pre>	Get the Network Information



### 3.1.2.3. Channel Information

Following table introduces about the 4 Channel Application of iinChip™.

Name	Description	Default Value
CH_TEST	Channel Information의	0xA5A5 – Valid
	Valid Check	Others – Invalid
CH_TYPE_X	No. "X" Channel's	Default - LB_TCPS
	Application Type	NOTUSE : Not Used
		DHCP_CLIENT : DHCP Client
		LB_TCPS : Loopb <mark>ack</mark> TCP Server
		LB_TCPC : Loopback TCP Client
		LB_U <mark>DP : Lo</mark> opback UDP
		WEB_SEVER : Web Server
CH_PORT_X	No. "X" Channel's	Little Endian
	Source/Destination	LB_TCPS : Default Source Port, 0x5000
	Port Number	LB_TCPC : Default Destination Port, 0x3000
		LB_UDP : Default Source Port, 0x3000
		WEB_SERVER : 80
CH_DESTIP_X	No. "X" Channel's	0xC0 A80003 (192.168.0.3)
	Destination IP Address	

#### <Table 3-9: Channel Information>

Channel Information is used for recording 4 Channel Application Types of iinChip™.

Channel Application Type includes Loopback TCP Server, Loopback TCP Client, Loopback UDP,DHCP Client, Web Server, and Channel Information is defined as APPTYPE enumeration type.

<table 3-10:<="" th=""><th>Channel Applicati</th><th>on Type&gt;</th></table>	Channel Applicati	on Type>
---	-------------------	----------

typedef enum _APPTYPE	
{	
NOTUSE,	
DHCP_CLIENT,	
LB_TCPS,	
LB_TCPC,	
LB_UDP,	
WEB SERVER	
, ,	



Channel Information is accessed as CHCONF Data Type.

### <Table 3-11: CHCONF Data Type Definition>

Type Definition	Global Instance
typedef struct _CHCONF	CHCONF ChConf;
<pre> u_int test; struct _CH_CONF {</pre>	

#### <Table 3-12: Channel Information Access Function>

Function	Description
void set_chconf(CHCONF* pChConf)	Save the Channel Information
void get_chconf(CHCONF* pChConf)	Ge <mark>t the C</mark> hannel Inform <mark>ation</mark>



# 3.2. EVB B/D Firmware

EVB B/D Firmware -EVB main()- can be divided into two parts. First, Manage Program that sets up various environments for running EVB B/D. And second, Loopback Programs that tests iinChip<sup>™</sup> performance and there are Internet Application using Internet Protocols such as DHCP, HTTP, DNS, and ICMP. Let's look at the sources list of which EVB B/D is composed and then look at each application source.



#### 3.2.1. Sources

#### <Table 3-13: EVB B/D Sources>

Classification	Files	Description
ann	ning ann h-ning ann c	Ping Request App implementation
upp	loopback h. loopback c	TCP LIDP Loophack Apps implementation
mcu	delay.n, delay.c	Delay Function – Walt_xxx()
	serial.h, serial.c	PIC UART control
	timer.h, timer.c	PIC Timer enable & disable
	types.h	PIC Data Type Definition
evb	channel.h, channel.c	Channel App Handler registration & cancellation
	config.h, config.c	EVB B/D Environment
	evb.h, evb.c	EVB B/D initialization
	lcd.h, lcd.c	EVB B/D Text LCD control
	led.h, led.c	EVB B/D LED(D1,D2) control
	manage.h <mark>,</mark> manage.c	Manage Ap <mark>p</mark>
inet	dhcp.h dchp.c	DHCP Client Protocol
	dns.h, dns.c	DNS Client Protocol
	httpd.h, httpd.c	HTTP Protocol
	ping.h, ping.c	Ping Protocol
main	main.h, <mark>ma</mark> in.c	EVB B/D F/W main()
rom	[webpage]	EVB B/D Web Pages
	romfs.h, romfs.c	EVB B/D Web Pages Image
	searchfile.h,searchfile.c	EVB B/D Web Page control
util	myprintf.h	printf() for debugging
	sockutil.h, sockutil.c	Utilities relating Socket
	util.h, util.c	Utilities
iinChip	iinchip_conf.h	System Dependant Defintion of iinChip™
	w3150a.h, w3150a.c	I/O Functions of iinchip™
	socket.h, socket.c	Socket APIs for iinChip™



### 3.2.2. How to Compile

Sources of <u>Chapter 3.2.1</u> compile in bundle after arranging SRC items.

Using MPLAB IDE, open the "EVBs" project. And, select "Make" or "Build All".

After compiling, the name that is assigned at Makefile TARGET and "Evbs.hex," the file that is assigned FORMAT are created. This can be loaded in PIC24.

### < Table 3-14 : iinChip™'s DEFINE Option >

#define LITTLE_ENDIAN	
#defineDEF_IINCHIP_DIRECT_M	10DE1
#defineDEF_IINCHIP_INDIRECT	_MODE 2
#defineDEF_IINCHIP_BUS	DEF_IINCHIP_DIRECT_MODE
//#defineDEF_IINCHIP_BUS	DEF_IINCHIP_INDIRECT_MODE

Since EVB B/D is Little-Endian system, LITTLE\_ENDIAN is defined and used. If the target system is Big-Endian, the defined items should be commented.

If iinchip<sup>™</sup> is intended to be used as different mode other than Direct Bus Mode, use desired Bus Mode Define as \_\_DEF\_IINCHIP\_BUS\_\_ instead of \_\_DEF\_IINCHIP\_DIRECT\_MODE\_\_. If DEFINE OPTION of iinchip<sup>™</sup> is changed the sources must Re-Build. To Re-Build project do "make clean" then "make".

### 3.2.3. How to download

For downloading the hex file, we use MPLAB IDE and MPLAB ICD 2.

- 1) Connect ICD Cable to J1 at the PM-PIC24 or J2 at the expension board.
- 2) Supply the power with EVB B/D.
- 3) Run MPLAB IDE.
- 4) Choose the "MPLAB ICD 2" in "Select Programmer" menu items.
- 5) Clicked "Program target device" button.

Please refer to userguide of MPLAB IDE for more information.

### 3.2.4. EVB B/D's main()

If we take closer look at main(), for certain amount of time, we wait for Manage Program from RS232 Terminal after initialization of board with board reset. At this point, if RS232 terminal displays the Manage Program entering command, EVB B/D environment such as network information and channel Information can be set and ping request program can be run.

If Manage Program is done or there is no entering command from RS232 terminal, the application for each of 4 channels of iinChip<sup>™</sup> is executed and initialized using previously set network information.

<Fig 3.3> process procedure of EVB B/D main(). Refer to "main/main.c"


If DHCP client exists in the application, the DHCP client obtains the network information from DHCP server by calling 'get\_IP\_DHCPS()' function. If DHCP client application does not exist or fails to obtain network information from DHCP server, the EVB B/D is initialized with previously-set network information.

After the initialization, it runs test applications of EVB B/D by calling each registered application handler. For further details on DHCP client program, refer to "Chapter 3.2.6.5 DHCP Client."



Function Name	Description	Location
int main(void)	EVB B/D main()	main/main.c
void evb_init(void)	PIC, Text LCD,	evb/evb.c
	UART initialization	
void net_init(void)	EVB B/D Network initialization	evb/evb.c
void check_manage(void)	Manage Program action wait and	evb/manage.c
	execution	
void register_channel_handler	Channel Application Handler registration	evb/channel.c
(u_char ch, void (*handler)(u_char))		
void unregister_channel_handler	Channel Application Handler cancellation	evb/ <mark>channe</mark> l.c
(u_char ch)		
void init_dhcp_client(SOCKET s, void	DHCP Client Program initialization	inet/dhcp.c
(*ip_update)(void),		
void (*ip_conflict)(void))		
u_int getIP_DHCPS(void)	Network Information acquisition from	inet/dhcp.c
	DHCP Server	
void check_DHCP_state(SOCKET s)	Check to expire the leased time from	inet/dhcp.c
	DHCP server	
void loopback_tcps(u_char ch)	Loopback - TCP Server	app/loopback.c
void loopback_tcpc(u_char ch)	Loopback - TCP Client	app/loopback.c
void loopback_udp(u_char ch)	Loopback - UDP	app/loopback.c
void web_server(u_char ch)	Web Server Program	app/webserver.c

## <Table 3-15: Reference Functions in EVB B/D's main()>







# 3.2.5. Manage Program

Manage Program is a program that sets up network and channel information by RS232 terminal and tests application with Ping Request to certain Destination.

Running the manage program can be started by call check\_manage() from main() function. check\_manage() checks if there is any entering command to Manage Program from RS232 terminal, that if character 'M' or 'm' is input or not. And if there is the command, it will enter to Manage Program through manage\_config(). If the user change the configuration, the EVB B/D automatically reboot and check\_manage() is skipped.









Function Name	Description	Location
void check_manage(void)	Decision of Manage Program is executed	evb/manage.c
	or not	
void manage_config(void)	Manage Program	evb/manage.c
u_char manage_network(void)	Configure Network Information	evb/manage.c
u_char manage_channel(void)	Configure Channel Information	evb/manage.c
u_char get_reset_flag(void)	EVB B/D's Auto/Manual Reset recognition	evb/config.h
	and confirm	evb/config.c
	Auto : SYSTEM_AUTO_RESET	
	Manual : S <mark>YS</mark> TEM_MANUAL_R <mark>ES</mark> ET	
void set_reset_flag(u_char flag)	Copy of EVB B/D Reset status	evb/config.c
void load_factory_netconf(void)	Factory Reset Network Information	evb/config.c
void load_factory_chconf(void)	Factory Reset Channel Information	evb/config.c
u_int uart_keyhit(u_char uart)	Checking the Input from UART(0,1)	mcu/serial.c
char uart0_getchar(void)	Read one character from UART0	mcu/serial.c
void uart0_putchar(char c)	Write one character to UART0	mcu/serial.c
void wait_1ms(u_int cnt)	Delay Function	mcu/delay.c
void ping_request(void)	Ping Request Test Program	app/ping_app.c

# <Table 3-16: Caller Function at Manage Program >

# 3.2.5.1. Network Configuration

Network Configuration is a sub-program of Manage Program and built with manage\_network(). And it's the program that sets up Network Information of EVB B/D. In general, MAC Address of Network Information is hardly updated after the initial setup. Accordingly, MAC Address setup does not provide Configuration Menu such as Source IP, Gateway IP, or Subnet Mask but it provides hidden menu. Also, MAC Address is not change at the time of Factory Reset. MAC Address is updated using 'M' or 'm'.







Function Name	Description	Location
u_char manage_network(void)	Configure Network Information	evb/manage.
		с
void get_netconf(NETCONF*	Get the Network Information that is	evb/config.c
pNetConf)	previously set	
void set_netconf(NETCONF*	Update the Network Information	evb/config.c
pNetConf)		
void display_netconf	Outputs the Network Information to	evb/config.c
(NETCONF* pNetConf)	the terminal	
Void load_factory_netconf(void)	Load Factory Reset Network	e <mark>vb/con</mark> fig.c
	Information	
char uart0_getchar(void)	Read one character from UART0	mcu/serial.c
void uart0_putchar(char c)	Write one character to UART0	mcu/serial.c
int uart_gets(u_char uart, char * str,	Read text lines from UART(0,1)	mcu/serial.c
char bpasswordtype, int max_len)		
void uart_puts(u_char uart, char * str)	Write text lines to UART(0,1)	mcu/serial.c
char VerifyIPAddress(char* src)	Check if the string is IP Address	util/sockutil.c
Unsigned long htonl	Transforms ordering of Long Type	util/sockutil.c
(unsigned long hostlong)	Data	
Unsigned long inet_addr	Transforms IP string into long type	util/sockutil.c
(unsigned char* addr)		

## <Table 3-17: Reference Functions in manage\_config()>

### 3.2.5.2. Channel Configuration

Channel Configuration, a sub-program of Manage Program is made of manage\_config() and decides which application to apply for each of 4 channels of iinChip™.

The application type that can be set up, there are DHCP Client, Loopback TCP Server/Client, Loopback UDP, and Web Server Program. Each Channel can be set up with any one of the applications above. But the DHCP Client can only be supported by the first channel and the setting cannot be repeated on other channel. TCP Server Program (LB\_TCPS,WEB\_SERVER) can be set repeatedly by channel and in such case the same port can be used. Here, the number of clients is as many as the same port number. Other application can be set repeatedly by channel but the same port number cannot use.







Function Name	Description	Location
u_char manage_channel(void)	Configure Channel Information	evb/manage.c
void select_ch_app	Select available Application Type and Setup	evb/manage.c
(CHCONF* pChConf, u_char ch)	required factors	
void get_chconf	Get Channel Information	evb/config.c
(CHCONF* pChConf)		
void set_chconf	Update Channel Information	evb/config.c
(CHCONF* pChConf)		
void display_chconf	Output Channel Information through Terminal	evb/config.c
(CHCONF * pChConf)		
void load_factory_chconf(void)	Factory Reset Channel Information	evb/config.c
char uart0_getchar(void)	Read one character from UART0	mcu/serial.c

#### < Table 3-19: Reference Functions in manage\_channel() >

#### 3.2.5.3. Ping Request Program

Ping Request Program is a program that sends Ping Request to certain destination. It uses ICMP protocol message on IP protocol and made with ping\_request().

ping\_request() is created with form similar to Ping program in DOS command prompt. It sends Ping request to destination after analyzing and processing the options.

Both domain name and IP address can be used as destination address for Ping request. In case of using domain name, domain name is changed into IP address using gethostbyname() or DNS. With the changed IP address the Ping request is sent.

When IP address is used with '-a' option, domain name can be obtained through gethostbyaddr() from DNS Server and the Ping request is sent to the IP address. When IP address is used without the '-a' option, Ping request is sent to input IP address without the connection with DNS.

gethostbyname(), gethostbyaddr() is DNS-related functions. For further information, refer to <u>Chapter 3.2.6.6</u> <u>DNS Client</u>. <Fig 3.8> and <Fig 3.9> are processing procedures of ping\_request().

<Fig 3.8> creates tokens of inputs of Command, Option, and Option Value and decides the related Bit of Argument Flag(PingArgsFlags).

<Fig 3.9> calls ping() based on relevant option and option after checking the validity of command, option, and option value with bits of argument flag. ping() sends Ping request message to certain destination and process the ICMP message which is received from any destination.











Let's take a brief look at Ping message before we proceed to Ping program.

Ping message has the value of '0'(Ping Reply) or '8'(Ping Request) at Type Field and Code Field of ICMP Message has 0. Also, Type Dependant Data Field(4Bytes) of ICMP Message can be re-defined as ID Field(2Bytes), Sequence Number Field(2Bytes) respectably. Data Field of ICMP Message is filled with the Ping Data to be looped back.

Finally, it calculates the checksum of ICMP header and Ping data of which the checksum fields are 0. After the calculation, it replaces 0 checksum fields with the newly calculated values.

<Fig 3.10> is a diagramming representation of the relationship between the ICMP Message Format and the Ping Message.



### <Fig 3.10: ICMP Message VS Ping Message>

Checking the Ping Reply about Ping Request can be done by checking if the values of ID, sequence number and ping data field are same. In case the Ping Reply does not come back in wait time, the ping can be sent again. In such case the Ping request is sent with the sequence number incremented by 1.

Ping Request message transmission and checking the Ping Reply message were done by ping(). The elements of ping() are Destination IP Address, Ping Reply Wait Time, number of Ping Requests, and Ping Data size and received Ping Replies are analyzed and processed to fit the elements.

<Fig 3.11> is the process of ping() and Ping message is defined and used as the Data Type of <Table 3-21>. Refer to "inet/ping.h"



#### <Table 3-20: PINGMSG Data Type Definition>

typedef struct _PIN	NGMSG	
{	_	
char	Type;	// 0 - Ping Reply, 8 - Ping Request
char	Code;	// Always 0
u_short	CheckSum;	// Check sum
u_short	ID;	// Identification
u_short	SeqNum;	// Sequence Number
char	Data[PINGBUF_LEN];	// Ping Data
}PINGMSG;		

Data field size of PINGMSG is 'PINGBUF\_LEN' Byte. PINGBUF\_LEN is defined '32'. But Data field Max size maybe '1472'. This is because the sending MTU of inChip™ is 1480 bytes and the sum of Code, CheckSum, ID, and SeqNum Field Size is 8 Bytes. If we subtract 8 from 1480 we get 1472. Hence, the size is 1472 bytes.

The results from ping() are saved in Data Type defined in <Table 3-22>.

#### <Table 3-21: PINGLOG Data Type Definition>

type	def structP	PINGLOG	
{			
	u_short	CheckSumErr;	
	u_short	UnreachableMSG;	
	u_short	TimeExceedMSG;	
	u_short	UnknownMSG;	
	u_short	ARPErr;	
	u_short	PingRequest;	
	u_short	PingReply;	
	u_short	Loss;	
}PIN	IGLOG;		

The saved Ping log can be output with RS232 terminal through DisplayPingStatistics() function. <Fig 3.12> shows the process procedures of DisplayPingStatistics().

CheckSumErr field is incremented by 1 whenever the check sum of Ping Reply from peer is received wrongly.

UnreachableMSG field and TimeExceedMSG field are incremented by 1 in case of receiving Unreachable Message or Time Exceeded Message from peer or gateway.

UnknownMSG field is incremented by 1 when the unknown message is received.

ARPErr field is incremented by 1 whenever ARP Reply is not received upon ARP Request to get the Hardware Address(MAC Address) of the peer.



PingRequest field is incremented by 1 whenever ping() sends Ping Request.

PingReply field is incremented by 1 whenever Ping Reply for Ping Request from the peer is received.

Loss field is incremented by 1 whenever Wait Timeout is occurred because nothing gets replied to the peer in certain period of time after sending Ping Request.











Ping Request program is, as explained previously, a program that uses ICMP Protocol which is running on IP Protocol. In case of using ICMP channel at iinChip<sup>™</sup>, as shown in <Fig 3.11> and <Fig 3.13>, which IP protocol to use must be decided. The socket must be created after calling setSn\_PROTO(s, IPPROT\_ICMP). IP RAW Channel must be created by calling socket() function when creating the socket. In case of closing ICMP Socket, setSn\_PROTO(s, 0x00) should be called after close(s) and clear the ICMP Flag which was set previously.







Function Name	Description	Location
void ping_request(void)	Ping Request program	app/ping_app.c
void ping_usage(void)	Outputs the instruction of Ping Request	app/ping_app.c
	program	
char ping	Sends Ping Request to specific	inet/ping.c
(int count, u_int size, u_int	destination, and processes ICMP	
time, u_char* addr, PINGLOG*	message received from any destination.	
log)		
void DisplayPingStatistics	Outputs the results from ping() calling	inet/ping.c
(PINGLOG log)		
void setSn_PROTO	Assigns IP protocol of the related socket	iinC <mark>hip/w3</mark> 150a.c
(SOCKET s, uint8 proto)		
char socket(SOCKET s,	Creates sockets related to as TCP/UDP/IP	iin <mark>Chip/s</mark> ocket.c
u_char protocol, u_int port,		
u_char flag)		
void close(SOCKET s);	Close the related socket	iinChip/socket.c
int sendto(SOCKET s,	Sends Datagram packet to specific	iinChip/socket.c
const u_char * buf, u_int len,	destination.	
u_char * addr, u_int port)		
int recvfrom(SOCKET s,	Rec <mark>eives</mark> Datagram packet from any	iinChip/socket.c
u_char * buf, u_int len,	destination	
u_char * addr, u_int * port)		
SOCKET getSocket(unsigned	Searches for socket has the designated	util/sockutil.c
char status, SOCKET start)	status	

# <Table 3-22: Reference Functions in ping\_request()>



# 3.2.6. Applications

It's a Network Application using iinChip<sup>™</sup> and it includes Loopback Program, Web Server, and DHCP Client. Application is selected by Manager Program.

# 3.2.6.1. Loopback TCP Server

The Loopback TCP Server program, EVB B/D works as server mode and AX1, the PC testing program works as client mode. AX1 tries to connect to EVB B/D and if the connection is successful, AX1 transmits the data stream through the TCP channel. EVB B/D returns back the data stream from AX1 without processing through the TCP Channel.

Loopback TCP Server Program uses loopback\_tcps() and <Fig 3.14> shows the process procedure of loopback\_tcps().





<table 3-23<="" th=""><th colspan="2"><table 3-23:="" functions="" in="" loopback_tcps()="" reference=""></table></th></table>	<table 3-23:="" functions="" in="" loopback_tcps()="" reference=""></table>	
Function Name	Description	Location
void loopback_tcps(u_char ch)	Loopback TCP Server program	app/loopback.c
u_char socket(SOCKET s,	Create the socket	iinChip/socket.c
u_char protocol, u_int port,		
u_char flag)		
u_char listen(SOCKET s)	It puts related Socket as server mode	iinChip/socket.c
u_int send(SOCKET s,	Transfer the data to the connected socket.	iinChip/socket.c
const u_char * buf, u_int len)		
u_int recv(SOCKET s,	Receive the data to the connected socket.	iinChip/socket.c
u_char * buf, u_int len)		
void disconnect(SOCKET s);	Close the connection of the socket.	iinCh <mark>ip/soc</mark> ket.c

If the server socket is in SOCK\_CLOSED status, loopback\_tcps() calls socket() with the elements of Sn\_MR\_TCP, Listen Port Number, and Option Flag to create TCP server socket.

The socket() function change the socket status as SOCK\_INIT regardless of the previous socket status. If the server socket is created successfully, it's put in TCP Server Mode after calling listen() with the server socket as the parameter. listen() makes the server socket status as listen status and maintains listen status until any client's connection.

At this point, when any client tries to connect to the server socket, the server socket status is changed from "listen" to "established". This is when the connection between Client and Server is complete and data transfer is possible in SOCK\_ESTABLISHED status.

Data is transferred using recv() and send() at the SOCK\_ESTABLISHED. The data transfer here is 1-on-1 transfer between EVB B/D(The server) and AX1(The client).

In the SOCK\_ESTABLISHED status if the client requests closing of the connection, the server socket status is changed from SOCK\_ESTABLISHED to SOCK\_CLOSE\_WAIT. In SOCK\_CLOSE\_WAIT status data communication is not possible and the server socket must be closed. In SOCK\_CLOSE\_WAIT status, disconnect() is called to close socket. disconnect() changes the socket status to SOCK\_CLOSED regardless of previous socket status.



## 3.2.6.2. Loopback TCP Client

At Loopback TCP Client program, EVB B/D works in client mode and AX1, PC test program, works in server mode. EVB B/D tries to connect to AX1 which is waiting as the server, if the connection is successful EVB B/D receives data stream through TCP channel and then EVB B/D sends back the received data stream to AX1 without processing.

Loopback TCP client program is created with loopback\_tcpc() and <Fig 3.15> is processing procedure of loopback\_tcpc().

If the client socket is in SOCK\_CLOSED status, loopback\_tcpc() calls socket() with the elements of Sn\_MR\_TCP, Any Port Number, and Option Flag to create TCP client socket.

In creating socket here, any port number is used for get\_system\_any\_port(). This is because connection may be failed if it tries to connect to the same server with same port number. After successfully creating the socket, call connect() with the elements of the client socket to connect to the AX1 server.

connect() makes the socket status into SOCK\_SYNSENT and keeps the status as SOCK\_SYNSENT until it receives the authorization for connection from the server. If the connection is successful the socket status is changed from SOCK\_SYNSENT to SOCK\_ESTABLISHED. In SOCK\_ESTABLISHED status the operation is same as explained for loopback\_tcps().





### <Fig 3.15: loopback\_tcpc()>

### <Table 3-24: Reference Functions in loopback\_tcpc()>

Function Name Description I		Location
void loopback_tcpc(u_char ch)	Loopback TCP Client Program	app/loopback.c
u_char socket(SOCKET s,	Related socket can be created as	iinChip/socket.c
u_char protocol, u_int port,	TCP/UDP/IP	
u_char flag)		
u_char connect(SOCKET s,	Attempts to connect to the specific	iinChip/socket.c
u_char * addr, u_int port)	server with related socket	
u_int send(SOCKET s,	Sends the data to related socket that is	iinChip/socket.c
const u_char * buf, u_int len)	in connection	
u_int recv(SOCKET s,	Receives the data to related socket that	iinChip/socket.c
u_char * buf, u_int len)	is in connection	
void disconnect(SOCKET s);	Close the related socket	iinChip/socket.c
u_int get_system_any_port(void)	Get any port number.	evb/config.c



#### 3.2.6.3. Loopback UDP

Loopback UDP Program is a program that uses unicast datagram communication of UDP protocol. It operates same as Loopback TCP Server/Client program does. UDP communication includes unicast datagram communication and broadcast datagram communication, and basically supports 1-to-many communication that is used for many destinations with one channel.

Loopback UDP program uses loopback\_udp() and <Fig 3-16> shows processing procedure of loopback\_udp().





Function Name	Description	Location
void loopback_udp(u_char ch)	Loopback udp program	app/loopback.c
u_char socket(SOCKET s,	Creates related socket as TCP/UDP/IP.	iinChip/socket.c
u_char protocol, u_int port,		
u_char flag)		
u_int sendto(SOCKET s,	Sends data to specific port of specific	iinChip/socket.c
const u_char * buf, u_int len,	destination related socket	
u_char * addr, u_int port)		
u_int recvfrom(SOCKET s,	Sends data to any port of any	iinChip/socket.c
u_char * buf, u_int len, u_char *	destination related socket	
addr, u_int * port)		
void close(SOCKET s)	Close related socket	iin <mark>Chip/so</mark> cket.c

# <Table 3-25: Reference Functions in loopback\_udp()>

If the udp socket is in SOCK\_CLOSED status, socket() is called using Sn\_MR\_UDP, Port Number, and Option Flag as the elements to create the UDP socket.

UDP communication, as opposed to TCP, is a datagram communication without the requirement of connection process. So, direct data communication is possible immediately after socket creation. After creation of UDP socket, the udp socket status will be changed from SOCK\_CLOSED to SOCK\_UDP. Here, not like TCP for data communication which uses send() and recv(), sendto() and recvfrom() are used.

This is because TCP is 1-to-1 communication method of which destination is known but UDP is 1-to-many communication without connection procedure. sendto() sends data to specific port of specific destination that is sent as an element, recvfrom() is used to receive the incoming data from temporary port. Destination information from recvfrom() is informed to user using destip and destport which are sent as elements. In loopback\_udp(), there is no example of using close(), but in case that the UDP communication is not needed anymore, close() can be called always and close the udp socket.

### 3.2.6.4. Web Server

Web Server program is a TCP server program using HTTP protocol which is used on TCP Protocol. Before building Web server program, message structure of HTTP protocol that is transmitted between Web server and Web client(Web browser) are needed to be understood.

HTTP, which stands for Hyper Text Transfer Protocol, is a protocol used in Internet for transferring between Web server and client browsers.



#### <Table 3-26: Web Browser's HTTP Request Operation Procedure >

Request of Client(Web Browser)

- --> URL Analysis(Transforming Domain Name to IP Address at DNS)
- --> Connection to server at the other end
- --> Client(Web Browser) requests document wanted from URL
- --> Sending Document(Server)/Receiving Document (Client)
- --> Displays received document on the browser

Web Server program analyzes method and URI(Uniform Resource Identifier) of HTTP Request message received from web browser, and in case the related URI simply requests for web page, then the page will be sent. If it requests an action such as CGI(Common Gateway Interface) then it takes the action and the result is informed in web page.

<Fig 3.17> shows HTTP message flow between web server and web client and <Table 3-28> shows structure of HTTP message.



Web Browser

<Fig 3.17: HTTP Message Flow>



	<table 3-27:="" format="" http="" message=""></table>	
HTTP-message	= Simple-Request	
	Simple-Response	
	Full-Request	
	Full-Response	
Full-Request	= Request-Line	
	*(General-Header   Request-Header   Entity-Header)	
	CRLF	
	[Entity-Body]	
Full-Response	= Status-Line	
	*((General-Header   Response-Header   Entity-Header) CRLF)	
	CRLF	
	[Entity-Body]	
Request-Line	= Method SP Reques <mark>t-URI SP HTT</mark> P-Version CRLF	
Status-Line	= HTTP-Version SP Status-Code SP Reason-Phrase CRLF	
Entity-Header	= Allow	
	Content-Encoding	
	Content-Length	
	Content-Type	
	Expires	
	Last-Modified	
	extension-header	
Entity-Body = *(	OCTET	
Method = "GET"	'   "HEAD"   "POST"   extension-method	

For further information on HTTP message, refer to RFC2616. HTTP request message varies depending on web browser. <Table 3-29> shows the examples of HTTP message communication between Internet Explores on Windows 2000 and EVB B/D.



	Request Message		
Ex1>	GET wiz_log.gif HTTP/1.1CRCF		
	Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/vndms-		
	powerpoint, application/vndms-excel, application/ms-word, */*CRCF		
	Accept Language: koCRCF		
	Accept Encoding: gzip, deflateCRCF		
	User-Agent: Mozilla/4.0 (compatible; MSIE 5.01; Windows NT 5.0; .NET CLR		
	1.3705)CRCF		
	Host: 192.168.0.2CRCF		
	Connection: Keep-AliveCRCF		
	CRCF		
Ex2>	GET http://192.168.0.2/LCDNLED.CGI?lcd=hi.+EVB B/D&led0=on HTTP/1.1CRCF		
	Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/vndms-		
	powerpoint, application/vndms-excel, application/ms-word, */*CRCF		
	Accept Language: koCRCF		
	Accept Encoding: gzip, deflateCRCF		
	User-Agent: Mozilla/4.0 (compatible; MSIE 5.01; Windows NT 5.0; .NET CLR		
	1.3705)CRCF		
	Host: 192.168.0.2CRCF		
	Connection: Keep-AliveCRCF		
	CRCF		
HTTP F	Response Message		
Ex1>	HTTP/1.1 200 OK CRCF		
	Content-Type: text/htmlCRCF		
	Content-Length: 1451CRCFCRCF		
	[Html Document]		
Ex2>	HTTP/1.1 200 OKCRCF		
	Content-Type: gif/imageCRCF		
	Content-Length: 613CRCFCRCF		

Web Server program is composed of web\_server() which manages HTTP server socket and proc\_http() which manages HTTP message.

<Fig 3.18> is processing procedure.





Since web\_server() is TCP server program is built in the similar way as loopback\_tcps() as explained in <u>Chapter 3.2.6.1</u>. Difference between web\_server() and loopback\_tcps() is in the data communication codes. web\_server() calls proc\_http() that processes HTTP request message from web browser at SOCK\_ESTABLISHED of the http socket.

After calling function proc\_http(), it waits until the HTTP Response message about HTTP Request from web browser, and then calls disconnect() to close the http socket.

This socket close is called Active Close and, in the case, EVB B/D requests the close to the client first. For your reference, Passive Close is where client requests disconnection first. The reason why web server program supports Active Close is that EVB B/D support to connect other client's connection.





<Fig 3.19: proc\_http()>

proc\_http() calls parse\_http\_request() to analyses the HTTP Request message received from web browser. If the METHOD of analyzed HTTP Request Message is "GET", "HEAD", or "POST", get\_http\_uri\_name() is called and URI Name is extracted from HTTP Request message. If extracted URI Name is "/" then replaces URI Name "/" to "index.html" which is web server default page of EVB B/D because this means that web



browser is requesting default page of web server.

After getting the HTTP Request Type of HTTP Request Message by calling find\_http\_uri\_type(), if HTTP Request Type is "CGI" then it performs the related CGI command process.

After processing CGI commands or in case that HTTP Request Type is not CGI, search file with URI Name from ROM File Image which is built in EVB B/D.

If the file is found, create HTTP Response message and send it.

HTTP Response message is composed of HTTP Response Header transmission and HTTP Response Body transmission. For transmission of HTTP Response Header, it calls make\_http\_response\_head() using HTTP Request Type as the element to create HTTP Response Header. After transmitting the created HTTP Response Header, since the HTTP Response Body is transmitted. For example, if the HTTP Response body is any file in ROM File Image, the files are much bigger than the MTU of iinChip<sup>™</sup>. Hence it has to be divided into maximum size of iinChip<sup>™</sup> before the transmission. At this point, if system environment variables that are defined in EVB B/D in HTTP Response Body are existent, it calls replace\_sys\_env\_value() and replaces system environment variables to system environment value that was stored in EVB B/D.

<Table 3-29: System Environment Variables Usage at "evbctrl.html" >

<font color="#FEFEEF"></font> Source IP
<input name="sip" size="20" type="text" value="\$SRC IP ADDRES\$"/>
<pre><font color="#EEEEE"> </font>Gateway IP</pre>
the wight = 77% singut paper = "win" type = "are" size = 20" value = "\$CW ID ADDESS\$" > /td>
the second
<font color="#FEFEEF"></font> Subnet Mask
<pre>ctd height="27"&gt;<input name="sn" size="20" type="text" value="\$SUB_NETMASK\$"/></pre>
<font color="#FEFEFE"></font> DNS Server IP
<pre><input name="dns" size="20" type="text" value="\$DNS_SERVER_IP\$"/></pre>
vidth="110" neight="22"> <toht color='#FEFEF"'></toht> width=width="20"> <toht color='#FEFEF"'></toht> width="20"> <toht color='#FEFEF"'></toht> width="20"> <toht color='#FEFEF"'></toht> width="20"> <toht color='#FEFEF"'></toht> width="20"> <toht color='#FEFEF"'><toht color="#FEFEF&lt;/toht">width="20"&gt;<toht color="#FEFEFEF&lt;/td"></toht></toht></toht></toht></toht></toht></toht></toht></toht></toht></toht></toht></toht></toht>
\$SRC_MAC_ADDRESS\$

<Table 3-30> is part of "evbctrl.html" in ROM File Image of EVB B/D.

The length of the system environment variables is defined to fit the length of system environment value to be replaced. For example, if Source IP Address of EVB is expressed in string the maximum is 16. Hence the length of \$SRC\_IP\_ADDRESS\$ is 16 as well. 'ROM File System' of EVB B/D can be created with "ROMFileMaker.exe" provided by WIZnet. Refer to "ROM File Maker Manual Vx.x.pdf" for further information.

HTTP Request message can be divided into Method and Request-URI by parse\_http\_request() and stored in



'st\_http\_request' Date Type which is defined in <Table 3-31>. It gets the requested URI Type with get\_http\_uri\_type().







<Fig 3.21: find\_http\_uri\_type()>

Request-URI which is saved in URI [MAX\_URI\_SIZE] of st\_http\_request has URI Name before "?" symbol and Query String after "?" sign. When Request-URI is transferred from Web Browser to Web Server, SP (Space) text is transmitted in the form of '+' and, other Reserved Texts are transmitted in the form of "%HEXHEX." Accordingly, Reserved Texts in Request-URI needs to be decoded to the previous value, from '+' to SP and from %HEXHEX to related ASCII vales. For the details of Request-URI decoding refer to RFC1738. URI name of Request-URI is extracted with get\_http\_uri\_name().Query String of Request-URI can include one or more "variable=value" pair that has "&" as a separator. Through function get\_http\_param\_value() it can extract the wanted variable value in Query String.





CGI processing of Web Server Program at EVB B/D is different from general Web Server Program which is based on OS. Web Server Program which is based on OS creates separate process to take case of communication between processes independently. However, Web Server of EVB B/D is OS-less, so, instead



of making independent process, it calls relevant functions to deal directly with CGI processing. EVB B/D supports "NETCONF.CGI" which updates Network Information and "LCDNLED.CGI" which controls text LCD, D1/D2 LED of EVB B/D. <Fig 3.23> and <Fig 3.24> shows both CGI processing.









Processing LCONLED, CGI

Extract the value of 'lcd'

≪param != NULI

N

N

Extract the value of 'led0'

param != NUL



<FORM> of NETCONF.CGI is submitted in "POST" Method. <FORM> submitted using "POST" Method is not submitted in Query String but submitted in Entity Body of HTTP Request Message. Such Value of Parameter for NETCONF.CGI, also, is used to extract related Parameter Value using get\_http\_param\_value().

<FORM>of LCDNLED.CGI is submitted in "GET" Method and <FORM> submitted as "GET" Method is submitted in Query String of Request-URI. Parameters submitted by Query String of Request-URI can also extract Parameter Value using get\_http\_param\_value().



Function Name	Description	Location
void web_server(u_char ch)	Web Server Program	app/webserver.c
void proc_http(SOCKET s,	Processes HTTP Message using related	app/webserver.c
u_char * buf, int length)	socket	
u_int replace_sys_env_value	Change Pre-defined System	app/webserver.c
(u_char* base, u_int len)	Environment Variables in HTTP	
	Response Message to Real Values.	
void parse_http_request	Analyzes and processes HTTP Request	inet/httpd.c
(st_http_request *, u_char *)	Message and saves it in st_http_request	
	structure.	
void find_http_uri_type	Gets MIME Type of HTTP Request	ine <mark>t/httpd</mark> .c
(u_ch <mark>ar *, ch</mark> ar *)	Message.	
char* get_http_uri_name	Gets Request-URI Name of HTTP	inet/httpd.c
(char* uri)	Request Message.	
char* get_http_param_value	Gets R <mark>e</mark> levant Para <mark>meter</mark> Value in	inet/httpd.c
(char* uri, char* param_name)	Query String of Request-URI	
void unescape_http_uri(char *	Transforms Escape Character	inet/httpd.c
url)		
void make_http_response_head	Creates header of HTTP Response	inet/httpd.c
(char *, char, u_long)	Message	
u_char socket(SOCKET s,	Creates related socket as TCP/UDP/IP	iinChip/socket.c
u_char protocol, u_int port,		
u_char flag)		
void listen(SOCKET s)	Puts the related socket in Server Mode	iinChip/socket.c
u_int send(SOCKET s,	Sends data using connected socket	iinChip/socket.c
const u_char * buf, u_int len)		
u_int recv(SOCKET s,	Receives data from the data from the	iinChip/socket.c
u_char * buf, u_int len)	connected socket	
void disconnect(SOCKET s)	Closes the connection of the socket	iinChip/socket.c
void replacetochar(char * str,	Changes the special characters in text	util/util.c
char oldchar, char newchar)	rows into new characters.	

#### <Table 3-31: Reference Functions in web\_server()>

3.2.6.5. DHCP Client

DHCP Client program is a program that assigns the network information from DHCP server in the network. Note that, If DHCP Client program must be started prior to other program because it manages Network


Information setup. First, review basic facts on DHCP(Dynamic Host Configuration Protocol) and get further into the usage of DHCP Client program.

DHCP uses UDP protocol in Transport Layer and communicates with DHCP server using broadcast of UDP. The reason why it uses broadcast is because it has no IP address and DHCP server IP address is unknown. When UDP broadcast at iinChip<sup>™</sup>, destination IP address needs to be set '255.255.255.255' for broadcast packet transmission.

<Fig 3.25> is a Message Flow between DHCP Server and Client.



#### <Fig 3.25: DHCP Message Flow>

First of all, DHCP client broadcasts DISCOVERY message to the local Network. If DHCP server exists at the network then DHCP server receives Discovery message and provides network Information such as IP, G/W IP, Subnet Mask, and DNS sever IP which can be used by DHCP Client, and information such as Lease Time to the DHCP Client as OFFER message. DHCP Client can detect DHCP server by receiving the OFFER message and then it sends REQUEST message to DHCP Server to use the information suggested by DHCP server. After receiving REQUEST message from DHCP Client, DHCP server finds out if the requested network information is usable. If it is, it sends ACK message, if not, NACK message is sent to DHCP Client. After receiving ACK message from DHCP server, DHCP Client uses the offered network Information. The network information is valid only for the Lease Time suggested by DHCP server. Hence, if DHCP Client wants to keep using the network information, it retransmits REQUEST message to DHCP server to maintain network information usually after half of the Lease Time. In this process, DHCP client can get same or new network information from DHCP server. In case that it received new network information, the new network information must be used.

Message between DHCP server and client has the format as in <Fig 3.26> with the size of 544 Bytes. Refer to document 'RFC1541' for detailed explanation for each field of DHCP message Format. op Field of the first byte decided Request/Reply, and fields after ciaddr is used to deliver network information, and options field of 312 byte is used to transmit message type or the information such as Client Identifier.







To take a brief look at the Option Field of DHCP Message, Options Field has the format of <Fig 3.27>, it contains Magic Cookie Field, a Lease Identification Cookie with the size of 4 Byte and Code Set ranged from Code 0 to Code 255. From Code1 to Code 254, codes are composed of pairs of {Code, Len, Value}, and Code 0 and Code 255 are composed of {Code} only. For further explanation of each Code of Options Field, refer to RFC1533.



## <Fig 3.27: DHCP Message's Option Field Format>



Code	Enumeration Type	Description		
0	padOption	used to cause subsequent fields to align on word boundaries		
1	subnetMask	specifies the client's subnet mask		
3	routersOnSubnet	a list of IP addresses for routers on the client's subnet		
6	dns	specifies a list of DNS servers available to the client		
12	hostName	specifies the name of the client		
50	dhcpRequestedIPaddr	request that a particular IP address be assigned by the server		
51	dhcpIPaddrLeaseTime	a lease time for the IP address		
53	dhcpMessageType	used to convey the type of the DHCP message		
54	dhcpServerIdentifier	the IP addre <mark>ss</mark> of the selected server		
55	dhcpParamRequest	request values for specified configuration parameters		
61	dhcpClientIdentifier	specify client unique identifier		
2 <mark>55</mark>	endOption	marks the end of valid information		

#### <Table 3-33: DHCP Message Option Code Definition>

In the Option Field of 312 Bytes, the unused bytes are denoted with 0's padding.

<Table 3-34> is defined as Enumeration Data Type in "inet/dhcp.h" and shows most common Option Codes that are used in DHCP Client Program.

Other codes that are not defined in <Table 3-34> are skipped from DHCP Client Program.

The operation of DHCP Client Program is displayed in EVB B/D's main(). Refer to <Fig 3.3: EVB B/D's main()>

First, set up the MAC address to be used by DHCP Client at the initialization. MAC address is unique address for all the devices in the network. MAC address is most basic address in Network communication and necessary information to recognize DHCP Clients in DHCP Server. For MAC Address of DHCP Client program, it sets up SRC\_MAC\_ADDR which is Global Variable of DHCP Client using the MAC Address of EVB B/D. By calling init\_dhcp\_client() after setup of SRC\_MAC\_ADDR, it can register two function to be called in case of collision of the IP received from DHCP Server and in case of renewal the IP from DHCP Server.

When calling init\_dhcp\_client(), if each function is not specified, set\_DHCP\_network() and proc\_ip\_conflict() of DHCP Client Program are registered respectively.





<Fig 3.28: init\_dhcp\_client()>

When network information is renewal or IP collision occurs, register evb\_soft\_reset() to run auto reset for EVB B/D.

Second, Network Information acquirement can be done through getIP\_DHCPS().





## <Fig 3.29: getIP\_DHCPS()>

getIP\_DHCPS(), it initializes iinChip<sup>™</sup> using setSIPR(),setSHAR(),and etc. And then it initializes 'dhcp\_state' variable as DHCP Client program state to 'STATE\_DHCP\_DISCOVER'. After the initialization, it calls send\_DHCP\_DISCOVER() to transfer a DHCP DISCOVERY message to DHCP server.

After transmitting DISCOVERY DHCP message, it initializes timer variables which are the leased time of network information received from DHCP Server by calling reset\_DHCP\_time() and uses 'DHCP Timer' for 1-sec interval using set\_timer(). After the initialization of DHCP\_Timeout with 0, it waits for DHCP Message to be received from DHCP Server, as long as the 'DHCP\_WAIT\_TIME' defines and, as many as it's defines at 'MAX\_DHCP\_RETRY.' While waiting for 'DHCP\_WAIT\_TIME x MAX\_DHCP\_RETRY' time it continuously checks if dhcp\_state is changed to STATE\_DHCP\_LEASED through check\_DHCP\_state().



STATE\_DHCP\_LEASED state represents the network information is obtained and means that getIP\_DHCP() is done successfully. If network information is not obtained from DHCP Server during the waiting time for 'DHCP\_WAIT\_TIME x MAX\_DHCP\_RETRY' check\_DHCP\_state() sets DHCP\_Timeout to 1. When DHCP\_Timeout is 1, getIP\_DHCPS() returns failure after releasing the DHCP Timer.

When it failed to obtain network information from DHCP server, EVB B/D sets network configuration using default network information or previous obtained network information.

<Table 3-35> is a definition of State, Timeout , and Retry Count of DHCP Client.

Define	Description	
#define STATE_DHCP_DISCOVER 1	DISCOVERY Transmission	
#define STATE_DHCP_REQUEST 2	OF <mark>FE</mark> R Receiving & REQUEST Transmission	
#define STATE_DHCP_LEASED 3	ACK Receiving, Acquiring Network Information	
#define STATE_DHCP_REREQUEST 4	After obtaining Network Information, REQUEST	
	Retransmission	
#define STATE_DHCP_RELEASE 5	RELEASE Transmission	
#define MAX_DHCP_RETRY 3	Number of Same DHCP Message Transmission, 3	
	times	
#define DHCP_WAIT_TIME 5	Waiting time for receiving DHCP Message, 5 sec.	

## <Table 3-34: DHCP Client State & Timeout Definition>

At getIP\_DHCP(), 'DHCP\_XID' is variable to set up xid Field of DHCP message in <Fig 3.26: DHCP Message Format>, it must be unique and maintained the same value until Lease Time of network information is expired. DHCP\_XID is fixed with '0x12345678' on here, but it's recommended to use random value.

Be advised to set Source IP Address as '0.0.0.0.' when initializing iinChip<sup>™</sup> for communication with DHCP server. You can use any IP address to set Source IP address of iinChip<sup>™</sup> but using '0.0.0.0' is better because '0.0.0.0' corresponds to Class A in IPv4 addressing and it's a Null IP address that is not actually used. For this reason, there is no chance for collision with other network.

For DHCP server to transmit UDP broadcast packet, note that Flag field MSB of DHCP message must be set 1. Refer to <Fig 3.26: DHCP Message Format>.

<Table 3-36> is a part of code that sets up Flag field

## <Table 3-35: DHCP Message Flag Field Setup>

#define DHCP_FLAGSBROADCAST	0x8000	
pRIPMSG->flags = htons(DHCP_FLAGSB	ROADCAST);	

Third, management of Network Information obtained from DHCP server can be performed by check\_DHCP\_state().<Fig 3.30> shows DHCP message flow due to DHCP client state change in the check\_DHCP\_state() process.



## <Fig 3.30: DHCP Message Flow by DHCP Client State>

check\_DHCP\_state() checks if there is DHCP message from DHCP server, it receives and analyze DHCP message. By types of analyzed DHCP message, if it's DHCP message that can be receivable, it changes to next state after it changes DHCP Client State as DHCP Message Flow of <Fig 3.30> indicates.





<Fig 3.31: check\_DHCP\_state()>



check\_DHCP\_state() processes correspondingly with DHCP client state through the series of processes shown in <Fig 3.31>. If we take a look at DHCP\_STATE\_LEASED state at check\_DHCP\_state(), the Lease Time received from DHCP server is finite, in case that half of the Lease Time passed, it sends DHCP\_REQEUST Message to DHCP Server and changes it as DHCP\_STATE\_REREQUEST after it backs up the source IP. As it continuously transmits DHCP\_REQUEST to the server, network information is maintained.





parseDHCPMSG() receives DHCP message from DHCP server, categorizes Type of DHCP Message, and saves network information. When performing check\_DHCP\_state(), check\_DHCP\_Timeout() is called in case that DHCP message is not received during the DHCP\_WAIT\_TIME or received DHCP message from DHCP server is not expected, to retransmit DHCP message to DHCP server. If the retransmission of DHCP message is repeated as much as MAX\_DHCP\_RETRY, it transmits DHCP\_DISCOVER message to DHCP server after it initializes all the variables to start the connection of DHCP server and DHCP message.

Function Name	Description	Location
void init_dhcp_client(SOCKET s,	Initializes DHCP Client	inet/dhcp.c
void (*ip_update)(void),		
void (*ip_conflict)(void))		
u_int getIP_DHCPS(void)	Obtains network information from	inet/dhcp.c
	the server	
void check_DHCP_state(SOCKET s)	Manag <mark>e</mark> s <u>network</u> information	in <mark>et/dhc</mark> p.c
	obtained from DHCP Server	
void set_DHCP_network(void)	Applies network information	inet/dhcp.c
	obtained from DHCP server to	
	iinC <mark>hi</mark> p™.	
char parseDHCPMSG	Ana <mark>ly</mark> zes and processes DHCP	inet/dhcp.c
(SOCKET s, u_int length)	message	
void check_DHCP_Timeout(void)	Retransmits the DHCP message	inet/dhcp.c
	when DHCP connection Timeout	
	occurs	
char check_leasedIP(void)	Check if the IP obtained from DHCP	inet/dhcp.c
	server is faced with collision.	
void reset_DHCP_time(void)	Initializes DHCP Timer related	inet/dhcp.c
	variables.	
void DHCP_timer_handler(void)	DHCP Timer Handler	inet/dhcp.c
void send_DHCP_DISCOVER	Transmits DHCP_DISCOVER	inet/dhcp.c
(SOCKET s)	message to DHCP server.	
void send_DHCP_REQUEST	Transmits DHCP_REQUEST	inet/dhcp.c
(SOCKET s)	message to DHCP server.	
void	Transmits	inet/dhcp.c
send_DHCP_RELEASE_DECLINE	DHCP_DISCOVER/DHCP_DECLIN	
(SOCKET s,char msgtype)	E message to DHCP server	
u_int init_dhcpc_ch(SOCKET s)	Creates DHCP client socket.	inet/dhcp.c

#### <Table 3-36: Reference Functions in DHCP Client>



u_char socket(SOCKET s,	Creates sockets as TCP/UDP/IP	iinChip/socket.c
u_char protocol, u_int port, u_char		
flag)		
u_int sendto(SOCKET s, const	Transmits data through specific port	iinChip/socket.c
u_char * buf, u_int len, u_char * addr,	of specific Destination	
u_int port)		
u_int recvfrom(SOCKET s, u_char *	Receives data through any port of	iinChip/socket.c
buf,	any destination.	
u_int len, u_char * addr, u_int * port)		
void close(SOCKET s)	Closes the Socket	iinChip/socket.c

## 3.2.6.6. DNS Client

Let's take a brief look at the DNS(Domain Name System) before DNS Client setup is introduced.

DNS is a system that transforms Internet Domain Name to Internet IP Address or Internet IP Address to Internet Domain Name. DNS is composed of Name Server that contains mapping table between IP Address and Domain Name, and DNS resolver that receives query results by transmitting query to Name Server.

DNS resolver queries IP address or Domain Name to be transformed to local Name Server. Local Name Server which received the Query searches its DB and answers back to the Resolver. If Resolver cannot find the information it looks up, Local Name Server sends the received query to Name Server at higher layer and the received answer can be sent to the Resolver.



#### <Fig 3.33: Domain Name System Structure & DNS Message Flow>

As seen in <Fig 3.33>, DNS Query and DNS Answer Message transmittable between DNS Resolver and Name Server are composed of 5 Sections in <Fig 3.34>.





<Fig 3.34: DNS Message Format>

Header Section has fixed 12 Bytes length and the other 4 sections have variable lengths. Answer, Authority, Additional Section other than Header and Question Section are called Resource Records(RRs). Each of Header, Question, and RRs has different format.







<Fig 3.37: Recode Resources Format>

Header Section of DNS Message holds Type of Message, DNS Query Type, and count information on variable length section.

In <Fig 3.35: Header Section Format>, QR Field gets 0 when DNS Message is a request from Resolver to Name Server and gets 1 when it's from Name Server to Resolver. Opcode Field gets 0 when it queries Domain Name as IP Address and gets 2 when it queries Name Server status.

QDCOUNT, ANCOUNT, NSCOUNT, and ARCOUNT Field, count information for variable length, represent Block Count that is composed of Question, Answer, Authority, and Additional Section. Question Section is made of Block of <Fig 3.36: Question Section Format> Recode Resources(RRs) which are Format. Answer, Authority, and Additional Sections are composed of Block of <Fig 3.37>.

For example, if QDCOUNT is 1, ANCOUNT is 10, NSCOUNT is 10, and ARCOUNT is 10 then Question Section is composed of Block 1 of <Fig 3.36: Question Section Format> and Answer, Authority, and Additional Section are composed of 10 Blocks in <Fig 3.37>.

NAME of <Fig 3.37>, QNAME Filed of <Fig 3.36> and RDDATA Field also get variable lengths. QNAME and NAME are variable length fields which are composed of <Fig 3.36> Format and they process each field. RDDATA, variable length field, processes using the data length of RDLENGTH Field.

For further details, refer to RFC1034 and RFC1035

DNS Message is operated by Data Type defined in <Table 3-38>. Refer to "inet/dns.h"



<table 3-37:="" dns="" messa<="" th=""><th>ge Data Type&gt;</th></table>	ge Data Type>
/* Header Section */	
typedef struct _DHDR	
{	/* Identification */
u char flag0:	
u_char flag1;	
u_int qdcount; /* Question count */	
u_int ancount; /* Answer count */	. */
u_int nscount; /* Authority (name server)	count ^/
BHDR:	/
/* Question Section */	
typedef struct _QUESTION	
// char* gname:	// Variable length data
u int gtype:	// valiable length data
u_int qclass;	
}DQST;	
/* Deserves Deserves */	
/* Resource Records */	
typedet struct _RESOURCE_RECORD	
{	
// char* _name;	// Variable length data
u_int _type;	
u_int _class;	
u long ttl;	
u int rdlen:	
// char* rdata:	// Variable length data
	in valiable length data
}DKK;	

DNS Resolver works based on gethostbyaddr() and gethostbyname(). gethostbyaddr() transforms Internet IP Address to Internet Domain Name and gethostbyname() transforms Internet Domain Name to Internet IP Address. gethostbyaddr() and gethostbyname() test the setup of DNS Name Server IP Address and search Free Channel of iinChip<sup>™</sup> needed for connection with DNS Name Server. If Free Channel of iinChip<sup>™</sup> exists, gethostbyaddr() and gethostbyname() call dns\_query() with 'BYNAME' or 'BYIP' as the elements. For examples of gethostbyaddr() and gethostbyname(), refer to <u>Chapter 3.2.5.3</u> Ping Request Program.

Actual connection with DNS Name Server is performed through dns\_query(), and gethostbyaddr() and gethostbyname() are reporting only the result of dns\_query().

## <Table 3-38: Query Type Definition at dns\_query()>

typedef enum \_QUERYDATA{**BYNAME**,**BYIP**}QUERYDATA; /\* Query type \*/





<Fig 3.38: gethostbyaddr() & gethostbyname()>

dns\_query() initializes the buffer that is needed for DNS inter-working and creates QNAME of Question Section based on Query Type 'BYNAME', and 'BYIP.' If the Query Type is 'BYNAME,' that is, when querying the Domain Name with IP Address, Domain Name can be used as QNAME without transformation.

When Query Type is 'BYIP,' that is, when querying the Domain Name with IP Address, change IP Address to IP Address String and QNAME is used after adding "**in-addr.arpa**" to the changed IP Address String. After the creation of QNAME, UDP Socket is created for DNS inter-working and DNS Request Message is created by calling dns\_make\_query(). If DNS Request Message is created successfully DNS Request Message is sent to DNS Name Server through UDP Socket. After sending DNS Request Message it receives DNS Response Message or waits until the waiting time is expired.

If DNS Response Message is received from DNS Name Server during the waiting time it analyzes received DNS Response Message using dns\_parse\_response().dns\_query() returns IP Address or Domain Name depending on Query Type.

<Fig 3.39> is dns\_query()'s process map









**EVB-PIC24 User's Manual** 



dns\_makequery() creates DNS Request message to be sent to DNS Name Server. Since DNS Request Message can query only with Header, Question Section, RRs Sections is not needed to be created. If you examine the header section creation at dns\_makequery(), first, it sets ID Field values as any value in DNS Message inter-working. On here, ID is set with 0x1122, and for further inter-working, the value is incremented by 1. QR, Opcode, AA, TC, RD Field are set as QR\_QUERY, OP\_QUERY/OP\_IQUERY, 0, 0, 1 respectively through MAKE\_FLAG0(), and RA, Z, RCODE Field are set as 0, 0, 0 respectively through MAKE\_FLAG1(). **Table 3-39: Constants and MACRO used in Header Section>** 

#define QR_QUERY	0				
#define QR_RESPONSE	1				
#define OP_QUERY	0 /* a standard query (QUERY) */				
#define OP_IQUREY	1 /* an inverse query (IQUERY) */				
#define OP_STATUS	2 /*a server status request (STATUS)*/				
#define MAKE_FLAG0(qr, op, aa, tc, rd)					
( ((qr & 0x01) << 7) + ((op & 0x0F) << 3) + ((aa & 0x01) << 2) + ((tc & 0x01) << 1) + (rd & 0x01) )					
#define MAKE_FLAG1(ra, z, rcode)					
(((ra & 0x01) << 7) + ((z & 0x07) << 4) + (rcode & 0x0F))					

Since the count fields, QDCOUNT, ANCOUNT, NSCOUNT, and ARCOUNT, have only one question each is set as 1, 0, 0, 0 respectively.

Let's look at Question Section. QNAME Field is Field that sets IP Address String. Domain Name and IP Address String are composed of Label length of 1 byte and Label of MAX 63 Byte. The end of QNAME is always set with 0 to find out the variable length of QNAME. <Fig 3.41> is actual example of transformation of Domain Name "www.wiznet.co.kr" in QNAME field.







QTYPE Field of Question Section is set 'TYPE\_PTR' when is holds Domain Name as QNAME. When it's IP Address it's set as 'TYPE\_A' and QCLASS Field is set as 'CLASS\_IN' since it is included in Internet. Table 3-41 is definition of constants that are used in QTYPE & QCLASS Fields.

Definition			Description	
#define	TYPE_A	1	The ARPA Internet	
#define	TYPE_NS	2	an authoritative name server	
#define	TYPE_MD	3	a mail destination (Obsolete - use MX)	
#define	TYPE_MF	4	a mail forwarder (Obsolete - use MX)	
#define	TYPE_CNAME	5	the canonical name for an alias	
#define	TYPE_SOA	6	marks the start of a zone of authority	
#define	TYPE_MB	7	a <mark>mailbox doma</mark> in nam <mark>e</mark>	
#define	TYPE_MG	8	a mail group member	
#define	TYPE_MR	9	a mail ren <mark>ame do</mark> main name	
#define	TYPE_NULL	10	a null RR	
#define	TYPE_WKS	11	a well known service description	
#define	TYPE_PTR	12	a domain name pointer	
#define	TYPE_HINFO	13	host information	
#define	TYPE_MINFO	14	mailbox or mail list information	
#define	TYPE_MX	15	mail exchange	
#define	TYPE_TXT	16	text strings	
#define	QTYPE_AXFR	252	A request for a transfer of an entire zone	
#define	QTYPE_MAILB	253	A request for mailbox-related records	
#define	QTYPE_MAILA	254	A request for mail agent RRs	
#define	QTYPE_TYPE_AL	L 255	A request for all records	
#define CLASS_IN 1			Internet	
#define CLASS_CS 2			CSNET class	
#define CLASS_CH 3			CHAOS class	
#define CLASS_HS 4			Hesiod [Dyer 87]	
#define QCLASS_ANY 255			Any class	

## <Table 3-40 : Constants Definition at QTYPE & QCLASS Field>







dns\_parse\_response() of <Fig 3.42> analyzes Response Message received by DNS Name Server. dns\_parse\_response() checks if it's same as Request Message ID that was sent to DNS Name Server and it also checks if the message received is a Response Message by checking QR Field of Header Section. If the received message is Response from DNS Name Server the success of change is decided by checking the RCODE Field value of Header Section.

<Table 3-42> is definition of constants that are used in RCODE Field.

Definition	Description		
#define RC_NO_ERROR 0	No error condition		
#define RC_FORMAT_ERROR 1	Format error - The name server was unable to interpret		
	the query		
#define RC_SERVER_FAIL 2	Server failure - The name server was unable to process		
	this <mark>qu</mark> ery due to a problem with the name server		
#define RC_NAME_ERROR 3	Nam <mark>e</mark> Error - M <mark>eaning</mark> ful only for <mark>respon</mark> ses from an		
	auth <mark>o</mark> ritative nam <mark>e serve</mark> r, this code signifies that the		
	domain name referenced in the query does not exist.		
#define RC_NOT_IMPL 4	Not Implemented - The name server does not support		
	the requested kind of query.		
#define RC_REFUSED 5	Refused - The name server refuses to perform the		
	specified operation for policy reasons.		

## <Table 3-41 : Constant Definition at Header Section's RCODE Field>

If the RCODE is RC\_NO\_ERROR then variable length sections such as Question, Answer, Authority, and Additional Section are analyzed. Since the necessary information is set in Answer Section, in this case, it's analyzed and processed up to Answer Section, and other section analysis and process are not performed. If you need information on Authority and Addition Section you can get them easily on your own.

Question Section is processed as many as QDCOUNT of Header Section by calling dns\_parse\_question(). Answer Section is processed as many as ANCOUNT of Header Section by calling dns\_parse\_question().





<Fig 3.43: dns\_parse\_question() & dns\_answer()>

dns\_parse\_question() analyses and processes Question Section. There are no information that actually used in the Question Section of DNS Request Message but it must be processed to get the starting position of Answer Section. Since QNAME Field of Question Section gets variable length, parse\_name() processes



QNAME Field to process the variable length processes and QTYPE, and QCLASS Field are skipped. dns\_answer() analyzes and processes Answer Section. Answer Section is a section where transformation actually takes effects and it performs appropriate process to TYPE Field of Answer Section.

TYPE of Answer Section has one of values from <Table 3-41 : Constants Definition at QTYPE & QCLASS Filed> and the value comes from either TYPE\_A or TYPE\_PTR. In case that the Domain Name is changed to IP Address, it can get the changed IP Address from TYPE\_A and if the IP Address is changed to Domain Name, Domain Name can be obtained from TYPE\_PTR. Changed Domain Name or IP Address are also processed and extracted by parse\_name().





QNAME, NAME, RDDATA Field are mostly composed as in <Fig 3.41: Example of QNAME Field transformation of Question Section >. However, it can be compressed to reduce DNS Message Size. Compression Scheme is expressed in 2 Byte. If the first byte, that is, the case where the upper 2 Bits are '11,' it means the Label is compressed. It has the offset that is composed of 1<sup>ST</sup> Byte excluding upper 2 bits and 2<sup>nd</sup> Byte

This offset is Offset of DNS Message and means the actual value of Label is located by the offset from the starting point of DNS message. When Compress Scheme tries to reuse Domain Name that was already used in DNS Message, relevant Domain Name sets the offset that is located in DNS Message as Indirect so that it can reduce the size of DNS Message. <Fig 3.45> is an example of Compress Scheme of DNS Message and its application.





"FOO.F.ISI.ARPA", "ARPA", and ROOT. "F.ISI.ARPA" is processed in the format of <Fig 3.41: Example of QNAME Field transformation of Question Section > with Offset 20 of DNS Message without compression.

In "FOO.F.ISI.ARPA," since the rest except for "FOO" is same as Name which is previously processed, "FOO" is processed with <Fig 3.41: Example of QNAME Field transformation of Question Section > Format without compression and the rest of names is processed by Offset 26. ROOT is the highest Domain and it's processed with Label Length Field of 0.

parse\_name(), before analysis of Name, checks if upper 2 bits of Label Length Byte are 11, if it's '11' the related Label analyzes the Label at the offset of DNS Message where the Label is located. If it's no '11' then the Label is analyzed and processed like as <Fig 3.41: Example of QNAME Field transformation of Question Section >.



Function Name	Description	Location
int gethostbyaddr	Changes IP Address to Domain Name	inet/dns.c
(u_long ipaddr,char* domain)		
u_long gethostbyname	Changes Domain Name to IP Address	inet/dns.c
(char* hostname)		
u_char dns_query	DNS Message Processing	inet/dns.c
(SOCKET s, u_long dnsip,		
u_char * domain_name,		
u_long* domain_ip,		
QUERYDATA querydata,		
u_int elapse)		
int dns_make_query	Creates DNS Request Message	in <mark>et/dns</mark> .c
(u_char op,char * qname)		
Int dns_parse_reponse(void)	Analyz <mark>es</mark> DNS Res <mark>ponse M</mark> essage	inet/dns.c
u_char * dns_parse_question	Analyzes Question Section of DNS	inet/dns.c
(u_char * cp)	Response Message	
u_char * dns_answer	Answer Section of DNS Response	inet/dns.c
(u_char *cp)	Message	
int parse_name(char* cp,char*	Analyzes NAME Field of Question,	inet/dns.c
qname, u_int qname_maxlen)	RRs Section	
u_char socket(SOCKET s, u_char	Creates sockets as TCP/UDP/IP	iinChip/socket.c
protocol, u_int port, u_char flag)		
u_int sendto(SOCKET s,	Transmits data through specific port of	iinChip/socket.c
const u_char * buf, u_int len,	specific Destination	
u_char * addr, u_int port)		
u_int recvfrom(SOCKET s,	Receives data through any port of any	iinChip/socket.c
u_char * buf, u_int len, u_char *	destination.	
addr, u_int * port)		
void close(SOCKET s)	Closes the related Socket	iinChip/socket.c

## <Table 3-42 : Reference Functions in DNS Client>



# 4. Hardware Designer's Guide

## 4.1. Block Diagram





## 4.2. Block Description

EVB B/D is composed of MB-EVB-PIC(EVB Base Board) and PM-PIC24(PIC MODULE).

Following 9 blocks are components of EVB B/D.

- PM-PIC24
- NM7010B<sup>+</sup>
- LCD
- SRAM
- RS-232 Port
- Expanded Board Interface
- Power Regulator
- 3.3V Power On System Reset

## 4.2.1. **PM-**PIC24

PM-PIC24 is composed of PIC Processor, 8MHz external crystal and header for interfacing to Base board(JP1,JP2), and ICD2(J1) Interface.



## <Fig 4.2: PM-PIC24 MODULE Dimension>

For easy development using EVB Board, all the port pin except for OSC(RC12, RC15) are connected to MB-EVB-PIC through module Interface(JP1, JP2). Pin description of Interface is shown in <Table 4-1: PM-PIC24



MODULE Pin Description>.

|--|

PM-				
PIC24	D:::: #	Din Nama	D:-	Description
MODULE	FIN #	Pin Name	Dir.	Description
Header #				
JP1	1, 2			3.3V power input
JP1	3	PGC		For development tools
JP1	4,5,6	RB1, RB2, RB3	I/O	Port B
JP1	7	LCD_RS	0	LCD RS signal.
JP1	8	LCD_E	0	LCD E signal.
JP1	9	/IINCHIP_IRQ	Ι	External interrupt pin
JP1	<mark>10,</mark> 11, 12	/WR	0	PMP bus control signal
		/RD		
		PMBE		
JP1	13	RB8	I/O	GPIO
JP1	14	D0(RE0)	I/O	PMP Data or PortE[0:7]
	~	~		
	21	D7(RE7)		
JP1	25	A9	I/O	PMP Address(alternatively I/O)
	~	~		
	31	A15		
JP1	33	/MCLR	Ι	Reset Signal Input process generated
				by EVB B/D's Reset Switch
JP1	40, 42	LED0, LED1	0	LED output
JP1	22, 31, 32,			Unused
	34, 35, 36,			
	37, 38, 39,			
	41, 43, 44,			
	46			
JP1	23, 24, 45,	GND		Signal Ground
	46, 47, 48,			
	49, 50			



PM- PIC24 MODULE Header #	Pin #	Pin Name	Dir.	Description
JP2	1, 2			3.3V power input
JP2	6	SCL	0	I2C Clock
JP2	7	SDA	I/O	I2C Data
JP2	8	RF0	I/O	Port F
JP2	9	PGD		For development tools
JP2	10	RF1	I/O	Port F
JP2	11	TX1	0	RS-232 Tx
JP2	13	RX1	Ι	RS-232 Rx
JP2	<mark>14</mark> , 17 ~ 24	A0, A1 ~ A8	0	PMP Address(alternatively I/O)
JP2	37 ~ 39, 41 ~	RD0 ~ RD2, RD6 ~	I/O	Port D
	44	RD9		
JP2	3 ~ 5, 12, 15,			Unused
	16, 27 ~ 32,			
	<mark>35, 3</mark> 6, 40,			
	45 ~ 48			
JP2	33	AVss		AVss
JP2	34	AVdd		AVdd
JP2	25, 26, <mark>49</mark> ,	GND		Signal Ground
	50			



#### PIC ICD2 and ICSP (J1) Pin Mapping



#### <Table 4-2: ISP Pin Description>

SIGNAL	Pin Number	Description
/MCLR	1	Reset
VCC	2	VCC
GND	3	Ground
PGD	4	Programming Data
PGC	5	Programming Clock
NC	6	Not connected

PIC ICD2 connector(J2) Pin map is same as above. Please refer to Chapter 9.3 in MPLAB ICD2 User's Guide.



## 4.2.2. NM7010B<sup>+</sup> MODULE

NM7010B<sup>+</sup> is a network module that is composed of W3150A<sup>+</sup>(TCP/IP hardwired chip), RTL8201CP(Ethernet PHY), and MAG-JACK(RJ45 with X'FMR).

TCP/IP,MAC protocol layer : W3150A<sup>+</sup> Physical layer : Ethernet PHY

Connector : MAG-JACK

For details on NM7010B<sup>+</sup> MODULE, refer to "NM7010B+ Datasheet Vx.x.pdf"

## 4.2.3. LCD

LCD is used for debugging and system status display. Pin Description of LCD Interface (JP5) is as follows.

## <Table 4-3: LCD PIN Description>

PIN#	EVB B/D PIN NAME/ LCD PIN NAME	DIR.	Description
1	GND / VSS		Signal Ground
2	5V / VDD	-	LCD Power Supply
3	V0 / V0	I	Voltage for LCD drive
4	LCD_RS/RS	I	Data/Instruction register select
5	GND / RW		Read/Write
6	LDC_E / E	I	Enable signal,start data read/write
7	RE0 / DB0	I/O	Data Bus Line
~	~		
14	RE7 / DB7		
15	NC1 / LED A	0	LED Anode, power supply+
16	NC2 / LED K	0	LED Cathode,ground 0V



## 4.2.4. SRAM

SRAM, with the size of 32Kbytes, is used as external data memory of PIC.

## 4.2.5. RS232 Port

It's a interface for Serial USARTs that is supported by PIC.

EVB B/D uses 9Pin DSUB male Type connector.

(PM-PIC24 use only one RS232 port.)

## 4.2.6. Expanded Board Interface

Expanded Board Interface is designed to be developed easily using EVB B/D. Most of the port pin of PIC24, power and many reserved pin are connected to Expanded Board Interface.

Bus Interface      66, 34, 67,    A0, A1, A2, A3, A4, A5,    O    PMP Address[0:15]      35, 68, 36,    A6, A7, A8, A9, A10,    PMP Address[0:15]    PMP Address[0:15]      69, 37, 70,    A11, A12, A13, A14,    PMP Address[0:15]    PMP Address[0:15]      73, 40, 74,    A15    PMP Data Bus[0:7]    PMP Address[0:15]      73, 40, 74,    PMP Data Bus[0:7]    PMP Data Bus[0:7]    PMP Bus Read Strobe      79,47,80,48    D5, D6, D7    PMP Bus Read Strobe    PMP Bus PMP Bus Vrite Strobe      81    SDA    I/O    I2C Bus Data Line      49    SCL    O    I2C Bus Clock Line      Port Interface    I/O    I2C Bus Clock Line      11    RB2    I    Port B      11    RB2    I    Port B      11    RB2    I    Port D      19    RD1    I/O    Port D      19    RD6    I/O    Port D	Pin #	Pin Name	Dir.	Description	
66, 34, 67,  A0, A1, A2, A3, A4, A5,  O  PMP Address[0:15]    35, 68, 36,  A6, A7, A8, A9, A10,  A11, A12, A13, A14,  A11, A12, A13, A14,    38, 71, 39,  A15  A15  A11, A12, D13, D14,    73, 40, 74,  A11, A12, D13, D4,  M  M    41  Image: Constraint of the state of	Bus Interface				
35, 68, 36, A6, A7, A8, A9, A10,  41, A12, A13, A14,    69, 37, 70,  A11, A12, A13, A14,    38, 71, 39,  A15    73, 40, 74,	66, <mark>34</mark> , 67,	A0, A1, A2, A <mark>3, A4</mark> , A5,	0	PMP Address[0:15]	
69, 37, 70,  A11, A12, A13, A14,    38, 71, 39,  A15    73, 40, 74,  A15    41  V    77,45,78,46  D0, D1, D2, D3, D4,  I/O    9,47,80,48  D5, D6, D7  PMP Data Bus[0:7]    53  /RD  O    76  PMBE  PMP BUS Enable    86  /WR  PMP Bus Write Strobe    81  SDA  I/O  I2C Bus Data Line    49  SCL  O  I2C Bus Clock Line    Port Interface  V/O  Port B  Interface    11  RB2  I/O  Port B    12  RB3  I/O  Port D    18  RD0  I/O  Port D    19  RD1  I/O  Port D    20  RD2  I/O  Port D	<mark>35, 6</mark> 8, 36,	A6, A7, A8, A9, A10,			
38, 71, 39, A15  A15    73, 40, 74, A1  A15    41  A1    77,45,78,46  D0, D1, D2, D3, D4, I/O  PMP Data Bus[0:7]    79,47,80,48  D5, D6, D7  PMP Bus Read Strobe    53  /RD  O  PMP Bus Read Strobe    76  PMBE  PMP BUS Enable    86  /WR  PMP Bus Write Strobe    81  SDA  I/O  I2C Bus Data Line    49  SCL  O  I2C Bus Clock Line    Port Interface  V/O  Port B  I1    10  RB1  I/O  Port B    11  RB2  I  I1    12  RB3  I  Port D    13  RD0  I/O  Port D    19  RD1  I  Port D    20  RD2  I  I    22  RD6  I  I	69, 37, 70,	A11, A12, A13, A14,			
73, 40, 74,  1  1  1    41  1  1  1    77,45,78,46  D0, D1, D2, D3, D4,  I/O  PMP Data Bus[0:7]    79,47,80,48  D5, D6, D7  1  1    53  /RD  O  PMP Bus Read Strobe    76  PMBE  PMP BUS Enable    86  /WR  PMP Bus Write Strobe    81  SDA  I/O  I2C Bus Data Line    49  SCL  O  I2C Bus Clock Line    Port Interface  1  RB2  1    10  RB1  I/O  Port B    11  RB2  1  1    12  RB3  1  1    13  RD0  I/O  Port B    14  RD1  I/O  Port D    19  RD1  I/O  Port D    19  RD2  I  I    20  RD6  I  I	38, 71, 39,	A15			
41  Image: Constraint of the system of th	<mark>73</mark> , 40, 74,				
77,45,78,46  D0, D1, D2, D3, D4, VO  PMP Data Bus[0:7]    79,47,80,48  D5, D6, D7  PMP Bus Read Strobe    53  /RD  O  PMP Bus Read Strobe    76  PMBE  PMP BUS Enable    86  /WR  PMP Bus Write Strobe    81  SDA  I/O  I2C Bus Data Line    49  SCL  O  I2C Bus Clock Line    Port Interface  VI  VI  PMP Bus Read Strobe    10  RB1  I/O  I2C Bus Clock Line    11  RB2  I  Port B    12  RB3  I/O  Port B    18  RD0  I/O  Port D    19  RD1  I/O  Port D    20  RD2  I  I/O  Port D	<mark>41</mark>				
79,47,80,48D5, D6, D7II53/RDOPMP Bus Read Strobe76PMBEPMP BUS Enable86/WRPMP Bus Write Strobe81SDAI/OI2C Bus Data Line49SCLOI2C Bus Clock LinePort Interface10RB1I/OPort B11RB2II12RB3IPort D18RD0I/OPort D19RD1II20RD2II22RD6II	<mark>77</mark> ,45,78,46	D0, <mark>D1, D2</mark> , D3, D4,	1/0	PMP Data Bus[0:7]	
53/RDOPMP Bus Read Strobe76PMBEPMP BUS Enable86/WRPMP Bus Write Strobe81SDAI/OI2C Bus Data Line49SCLOI2C Bus Clock LinePort Interface10RB1I/OPort B11RB212RB3I/OPort D19RD1I/OPort D20RD2II22RD6II	<mark>79,</mark> 47,80,48	D5, D6, <mark>D7</mark>			
76PMBEPMP BUS Enable86/WRPMP Bus Write Strobe81SDAI/OI2C Bus Data Line49SCLOI2C Bus Clock LinePort Interface10RB1I/OPort B11RB2II12RB3II18RD0I/OPort D19RD1II20RD2II22RD6II	53	/RD	0	PMP Bus Read Strobe	
86/WRImage: Minipage:	76	PMBE		PMP BUS Enable	
81SDAI/OI2C Bus Data Line49SCLOI2C Bus Clock LinePort Interface10RB1I/OPort B11RB2II12RB3II18RD0I/OPort D19RD1II20RD2II22RD6II	86	/WR		PMP Bus Write Strobe	
49SCLOI2C Bus Clock LinePort Interface10RB1I/OPort B11RB2II12RB3II18RD0I/OPort D19RD1II20RD2II22RD6II	81	SDA	I/O	I2C Bus Data Line	
Port Interface      10    RB1    I/O    Port B      11    RB2    -    -      12    RB3    -    -      18    RD0    I/O    Port D      19    RD1    -    -      20    RD6    -    -	49	SCL	0	I2C Bus Clock Line	
10RB1I/OPort B11RB212RB318RD0I/OPort D19RD120RD222RD6	Port Interface				
11RB2Image: Constraint of the second s	10	RB1	I/O	Port B	
12    RB3    I      18    RD0    I/O    Port D      19    RD1    I    I      20    RD2    I    I      22    RD6    I    I	11	RB2			
18    RD0    I/O    Port D      19    RD1    I    I      20    RD2    I    I      22    RD6    I    I	12	RB3			
19  RD1  Image: Constraint of the second	18	RD0	I/O	Port D	
20 RD2 22 RD6	19	RD1			
22 RD6	20	RD2			
	22	RD6			

#### <Table 4-4: Expanded Board Interface Pin Description>



	1	1	
1	RD7		
84	RD8		
85	RD9		
90	RF1	I/O	Port F
ETC			
4	LED1	0	LED pin
5	LED2		
13	LCD_RS	0	LCD RS signal
14	LCD_E	0	LCD E signal
15	PGC		Programming Clock
16	PGD		Programming Data
23	/IINCHIP_IRQ	I	External interrupt 0
33	/MCLR	I	Reset signal
91	TX1	0	RS-232 signals
92	RX1	1	
2, <mark>3, 6,</mark> 17,			Not Available(Do not use)
2 <mark>1, 25</mark> , 26,			
27, 28, 29,			
43, 50, 51,			
52, 54, 56,			
<mark>57</mark> , 58, 59,			
60, 61, 75,			
<mark>83</mark> , 87, 89,			
93, 94			
Power Interface	e		
31,32	5V	0	5V Power Supply
63,64	3.3V O		3.3V Power Supply
7	AVdd		AVdd
8	AVss		AVss
9,24,30,44,	GND		Ground
55,62,65,72,			
82,88			

Expanded Board Interface Connector, which is "PCN10BK-96S-2.54DS" of Hirose co., is a Din Connector 96Pin Female Rightangle Type. Connector of Male Type that is mated here is "PCN10-96P-2.54DS."



## 4.2.7. ICD2 Port

It's a interface for ICSP & ICD that is supported by MICROCHIP. EVB B/D uses 6Pin RJ11 connector. Please refer to Chapter 9.3 in MPLAB ICD2 User's Guide.

## 4.2.8. Power Regulator

EVB B/D gets 5V DC power through power adaptor. The powers used inside the board are 5V and 3.3 V. The regulator is LM1117MPX-3.3(U2).

## 4.2.9. 3.3V Power On System Reset

Manual reset and Power On Reset is implemented using RC analog circuit.


# 4.3. Schematic

4.3.1. MB-EVB-PIC

Please refer to " $\ensuremath{\mathsf{MB-EVB-PIC\_Schematics.PDF}\xspace"$  in the CD.

4.3.2. PM-PIC24

Please refer to "PM-PIC24\_Schematics.PDF" in the CD.

## 4.3.3. NM7010B<sup>+</sup>

Please refer to "NM7010B+\_Schematics.PDF" in the CD.



# 4.4. Parts List

4.4.1. MB-EVB-PIC Parts List Please refer to "**MB-EVB-PIC\_Partlist.PDF**" in the CD.

4.4.2. PM-PIC24 Parts List

Please refer to "PM-PIC24\_Partlist.PDF" in the CD.

#### 4.4.3. NM7010B<sup>+</sup>

Please refer to "NM7010B+\_Partlist.PDF" in the CD.



# 4.5. Physical Specification

### 4.5.1. Power Consumption

Power consumption of each component of EVB B/D is as in the following table.

#### < Table 4-5 EVB B/D Power Consumption >

Power Level	MIN	TYP	MAX	UNIT
5V	-	210	-	mA
3.3V		<mark>11</mark> 0	-	mA

Total Power consumption is 210mA X 5V = 1.5 Watt.